**Noname manuscript No.**
(will be inserted by the editor)

# A Deep Reinforcement Learning Based Algorithm for Reliability-Aware Multi-domain Service Deployment In Smart Ecosystems

**Godfrey Kibalya[1] · Joan Serrat[1] · Juan-Luis Gorricho[1] · Dorothy Okello[2] · Peiying Zhang[3]**

Received: date / Accepted: date

**Abstract** The transition towards full network virtualization will see services for smart ecosystems including smart metering, healthcare and transportation among others, being deployed as Service Function Chains (SFCs) comprised of an ordered set of virtual network functions. However, since such services are usually deployed in remote cloud networks, the SFCs may transcend multiple domains belonging to different Infrastructure Providers (InPs), possibly with differing policies regarding billing and Quality-of-service (QoS) guarantees. Therefore, efficiently allocating the exhaustible network resources to the different SFCs while meeting the stringent requirements of the services such as delay and QoS among others, remains a complex challenge, especially under limited information disclosure by the InPs. In this work, we formulate the SFC deployment problem across multiple domains focusing on delay constraints, and propose a framework for SFC orchestration which adheres to the privacy requirements of the InPs. Then, we propose a Reinforcement Learning (RL) based-algorithm for partitioning the SFC request across the different InPs while considering service reliability across the participating InPs. Such RL based algorithms have the intelligence to infer undisclosed InP information from historical data obtained from past experiences. Simulation results, considering both online and offline scenarios, reveal that the proposed algorithm results in up to 10% improvement in terms of acceptance ratio and provisioning cost compared to the benchmark algorithms, with up to more than 90% saving in execution time for large networks. In addition, the paper proposes an enhancement to a state of the art algorithm which results in up to 5% improvement in terms of provisioning cost.

**Keywords** Multi-domain orchestration · Service Function Chaining · Service Reliability · QoS Embedding · Multi-atrribute embedding

✉ Godfrey Kibalya and Peiying Zhang
Godfrey.mirondo.kibalya@upc.edu,zhangpeiying@upc.edu.cn
[1] Dept. Network Engineering, Universitat Politecnica de Catalunya, C/ Jordi Girona, 1-3 - Edif.C3 - Campus Nord - 08034 Barcelona - Spain
[2] Dept. Electrical and Computer Engineering, Makerere University, Kampala, Uganda
[3] College of Computer Science and Technology, China University of Petroleum (East China), Qingdao 266580, P.R. China

## 1 Introduction

In traditional networks, network functions such as firewalls and proxies are implemented by middle-boxes coupled with the hardware [1],[2]. This limits the service delivery of those networks and inhibits the infrastructure providers from optimally using the network resources in a dynamic manner [3], [4], [5]. In this regard, network virtualization has emerged as a promising technique to overcome this barrier by enabling the softwarisation of network functions [6], [7],[8]. As a result, services are instantiated as service chains consisting of an ordered set of Virtual Network Functions (VNFs) commonly referred to as Service Function Chains (SFCs), that can be dynamically deployed and scaled according to the real-time requirements [9], [10], [11], [4].

In practice, in smart ecosystem services such as smart metering, e-health and transportation systems, the geographical location at which the data is generated or measured may be different from that at which such data is processed, accessed or utilized [12], [13]. For-example, Fig. 1 shows an IoT system in which information from the medical and traffic networks is relayed to the core data-center (with possibility that there are

multiple core cloud servers with each belonging to different InPs) through an access network (wireless and optical) comprised of multiple InPs. Moreover, the location dependencies of certain network functions, coupled with the limited footprint of InPs may necessitate the deployment of SFCs of such services across substrate infrastructure belonging to multiple providers [14], [15], [16]. In this case, the end-to-end service is realized as a concatenation of service instances supported by different InPs. Given the criticality of future services, incorporating service reliability in the service deployment decision becomes inevitable under such a scenario, since a failure within a single InP may affect the performance of the entire service chain [17], [9], [18]. Moreover, softwarization increases the prospects of service failure as a result of software bugs, physical server failures and function malfunctions among others [19],[20],[21]. This presents a novel challenge regarding how to effectively and efficiently deploy the customised SFCs onto a multi-domain infrastructure with exhaustible resources while meeting the stringent service constraints such as delay and reliability. The problem is further exacerbated by the reluctance of the InPs to disclose some information related to their network topology and resources, which renders the single domain approaches such as those proposed in [22],[23],[19],[24], inappropriate for this problem [15].

The problem of multi-domain service deployment has been addressed in [25],[1],[26],[27],[28],[14],[29]. However, such heuristic approaches are not well suited for scenarios in which the service deployment decision requires to jointly consider multiple network attributes such as cost, delay and reliability among others. In this regard, machine learning (ML) based approaches have proved efficient in dealing with multi-attribute features, since these can effectively infer the influence of each attribute towards the output [30], [31],[32],[33],[34]. Moreover, by using ML, it is possible to exploit historical data based on previous service deployment to infer attributes that may not have been disclosed by the different InPs in order to guide future deployment decisions.

Generally speaking, the multi-domain service deployment problem can be seen to be composed of three major steps [35], [14]: 1) The resource matching step which involves identifying the domains that can potentially serve the request based on the disclosed information; 2) The request splitting step which involves selecting the optimal domains to host the request from all the above potential InPs inline with the deployment objective; 3) The binding step which involves reservation of resources and establishing the end-to-end service. The request splitting step involves a large number of possible combinations, hence, obtaining an optimal partitioning

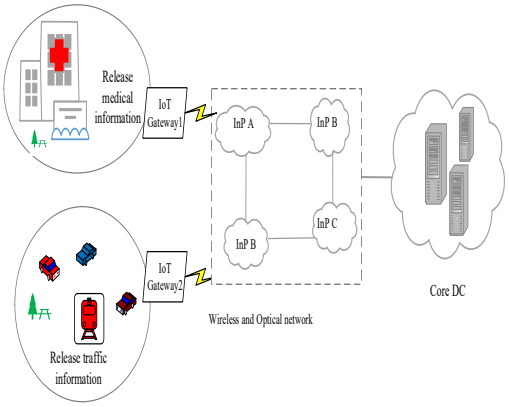



**Fig. 1** An illustration of multi-domain service deployment

decision is computationally complex. Therefore, in this work, we delegate this task to the RL-based module which is able to obtain a near-optimal solution in feasible time. In [9], we adopted ML to solve the problem of network slicing across multiple domains considering the case of full information disclosure with the objective of minimizing the provisioning cost. Different from the above work, in this work, we address the service deployment problem from a QoS perspective by incorporating service reliability in the deployment decision. Moreover, this work preserves the privacy of the different InPs. In addition, to deal with the common problem associated with reinforcement learning of invariable dimension of the input states, we adopt the concept of dummy features, so as to work with substrate networks of different sizes from the one used for training of the RL policy neural network. To the best of our knowledge, this is the first work incorporating service reliability in the problem of cross-domain service deployment.

In summary, our contributions are the following:

1. We propose a multi-domain orchestration framework that preserves the privacy of the different domains. The proposed framework is compatible with the ETSI management and orchestration framework proposed in [36], in which, each domain has an NFV orchestrator for intra-domain resource management.
2. We propose policy gradient based algorithm for selecting the optimal InPs/domains for the deployment of the SFCs, in order to efficiently share the underlying substrate resources while maximizing the InP revenue.
3. We propose enhancements to a state-of-the-art distributed service embedding algorithm, DistNSE, proposed in [29]. The proposed enhancements that are discussed in section 6.1 result in upto 10% improvement in terms of provisioning cost in some scenarios.
4. In order to assess the performance of the proposed algorithm, we perform extensive simulations, con-

sidering both online and offline scenarios. From the simulation results, the proposed algorithm is found to be scalable when considering an increasing size of the tested networks and requests.

The rest of the paper is organized as follows: Section 2 presents the related work. The proposed multi-domain orchestration framework is proposed in section 3. The network and service model, and the multi-domain service deployment problem description is presented in section 4. The proposed multi-domain service deployment algorithm is presented in section 5. The performance evaluation of the proposed multi-domain service deployment algorithm is presented in section 6 and the paper is concluded in section 7.

## 2 Related work

In the event that a single InP is not able to satisfy all the constraints associated with a given SFC request, the problem can be addressed by allocating the different segments of the request to different InPs; and, subsequently, concatenating those into a single end-to-end service. The problem of service chain deployment considering a single substrate network has been addressed in [22]-[24]. However, such approaches rely on full knowledge of the network topology, which is not possible in a realistic multi-domain scenario. Multi-domain service deployment has been addressed in literature either as a virtual network embedding (VNE) problem or as a service function chain placement (SFCP) problem. Within the context of our work, the existing works in these areas can be classified along three major lines based on: 1) The optimality of the solution, hence, exact and heuristic algorithms. Exact algorithms such as those in [27],[37],[38] result in optimal mapping solutions but at the expense of high execution time complexity. On the-other hand, heuristic approaches such as those in [28],[29] obtain near-optimal solutions with feasible time complexity; 2) The number of attributes jointly considered in computing the mapping solution, hence, single attribute and multi-attribute algorithms; In single attribute algorithms such as [4] , the mapping objective (e.g mapping cost minimization) is only influenced by a single attribute usually the amount/cost of resources that are allocated to the request. Therefore, by optimizing the considered attribute, the mapping objective is guaranteed to improve. On the-other hand, the mapping objective of multi-attribute approaches is jointly influenced by multiple attributes such as both the cost of the resources assigned to the request and the reliability of those resources. In this case, optimizing one attribute does not guarantee an improvement in the mapping objective; 3) The level of information disclosure considered, hence, full information disclosure (FID) and partial information disclosure (PID). FID approaches such as [39], [27],[40], [41],[42] consider the internal information within each InP to be visible to other InPs. In realistic situations however, due to reasons related to security and business competition, InPs are reluctant to share their topological and internal policy information with external entities. On the-other hand, PID algorithms such as [43],[4], [29] consider a limited level of information disclosure between the different InPs.

The works adopting approaches based on exact solutions to map the requests across different InPs with the objective of minimizing provisioning costs are found in [27],[37],[38], [15], [14]. However, exact approaches are associated with high time complexity, hence, not suited for delay sensitive applications that are typical of 5G and future networks. As a result, the work in [28],[29], [1], [44] adopt heuristic approaches for multi-domain SFC deployment considering a case of limited information disclosure. In [28], a distributed algorithm is proposed in which, upon the arrival of a request, the centralized orchestrator forwards the request to the different participating InPs. Then, following its internal policies, each InP selects the sub-SFC it can map internally. The results of the intra-domain mapping are then forwarded to the orchestrator which selects the optimal InPs for hosting the request with the goal of minimizing provisioning cost. In [29], the algorithm uses the exposed boarder nodes to compute all the feasible paths from source to destination. Then, for each of these paths, the first InP on the path receives the SFC request and selects a sub-SFC to bid for and forwards this and the SFC to the next InP along the path. Then, the receiving InP also selects a sub-SFC among the non-selected VNFs and also tries to compete for the sub-SFC selected by the previous InP. This process continues until the last InP along the path selects or competes for a sub-SFC of the request. Then, the path for mapping the request is chosen as the one which results in the least cost among all candidate paths. The work in [1] adopts a similar approach by using the exposed boarder nodes to obtain feasible abstracted paths connecting the source node to the destination node. The algorithm, then, partitions the SFC according to two approaches, namely, according to the number of domains crossed by the abstracted path, with the goal of minimizing request delay, and according to the available physical resources of the different domains constituting the abstracted paths. The authors of the above work adopt a similar approach in [45] and [44] with the goal of minimizing energy consumption. However, although these works reveal promising results in their performance, they are not suited for cases where

the mapping objective is jointly influenced by multiple attributes. Moreover, in the event that some of those attributes are not explicitly revealed by the participating InPs, such heuristic approaches do not have the desired intelligence to infer such undisclosed attributes from past performance experiences. While considering full information disclosure, the work in [39] proposes a node ranking approach for embedding of virtual requests across multiple domains. Such node ranking approaches have the ability to consider multiple attributes to compute the score/rank of anode towards mapping a request. However, besides the fact that such schemes cannot coordinate link and node mapping, obtaining the influence of each attribute towards the mapping objective using such a scheme is complex, since such influences may be dynamic in nature.

The above challenges can be overcome by using intelligent algorithms based on ML. The work in [46],[30],[47],[48] apply ML based techniques to the single domain VNE problem with the goal of minimizing provisioning costs. However, in a single InP scenario, the entire topological information is assumed to be known. Moreover, the entire traffic is subjected to the same InP policies in terms of service level agreement (SLA) and QoS from source to destination. The work in [49] incorporates reinforcement learning to the multi-domain problem. However, just like the above single domain works, the intelligent components in this work are the intra-domain orchestrators of the different InPs. Therefore, the centralized orchestrator cannot benefit from the past experiences in selecting the InPs to which to allocate the request, since the InPs may have conflicting objectives with the centralized orchestrator.

## 3 Proposed multi-domain orchestration framework

The multi-domain SFC orchestration framework considered in this work consists of three main players, as shown in Fig. 2: the tenant; a master orchestrator (MO); and a number of domain specific orchestrators (DSOs). The tenant is the initiator of a service request, with given specifications and constraints. This could be a service provider, a mobile virtual network operator or a vertical user, among others. The tenant specifies the request in terms of the topology, required resources (e.g.: computation, storage, memory, bandwidth) for the different VNFs and the corresponding interconnecting links, providing any other complementary constraint as well. The master orchestrator is the orchestrator corresponding to the domain from which the request is initiated. This entity is responsible for, among other functions: i) Mapping the specifications of the request to the global information provided by the different administrative domains, with the goal of identifying the potential domains for hosting the SFC request; ii) Invoking the intelligent DRL based algorithm to split the request among the different feasible domains in order to maximize the long term revenue; iii) Guiding the DSOs regarding the specific peering nodes on which to terminate the resource reservations during their respective intra-domain mapping stages; iv) Communicating the allocation results to the tenant, and the management of the life-cycle of the request in case of a successful provisioning. In this work, the DSOs always refer to all the orchestrators different from the one from which the request was initiated. Each DSO is supposed to have a complete view of its domain, including the internal topology, amount and type of resources, and QoS guarantees from the different resources. In case of a centralised approach, the MO can adopt the role of a Federation Manager as proposed in the frameworks of [50],[51].

The sequence diagram of our proposed framework is shown in Fig. 2, with blocks A-D indicating the key phases of the framework. Block A corresponds to a continuous phase, in which the different DSOs periodically advertise their public information, that is stored in an information repository (IR) for each domain. In order to adhere to the privacy requirements of the different domains, we limit each domain to only disclose the type of resources/functions that can be provisioned within that domain[45],[44]. However, information regarding the amount, location, and topology of those resources is presumed to be private, hence, undisclosed. Consequently, with respect to a given InP m, the master orchestrator observes a tuple of global information denoted by $< T^m, B^m, G^U >$ where $T^m$ is a set denoting the type of resources that can be provisioned within domain $m$, $B^m$ denotes the span or the geographical bounds in which the InP $m$ operates, and $G^U$ is the inter-domain connectivity graph comprised of the peering nodes and inter-domain links, including their respective attributes.

Block B corresponds to the candidate extraction phase. The phase is executed upon the arrival of a request from the tenant (event 2). The request is specified as a tuple $< G_v, \tau, \phi >$ where $G_v$ is a graph specifying the topology of the request and capturing the attributes and constraints of the VNFs and inter-connecting links, including the amount and type of resources to be provisioned over these VNFs and links. The parameter $\tau$ captures the life-time of the request. The parameter $\phi$ captures any other tenant-specific constraints such as maximum available budget, domain preferences and QoS, among others. On receiving the request, the MO must identify potential domains that may serve the

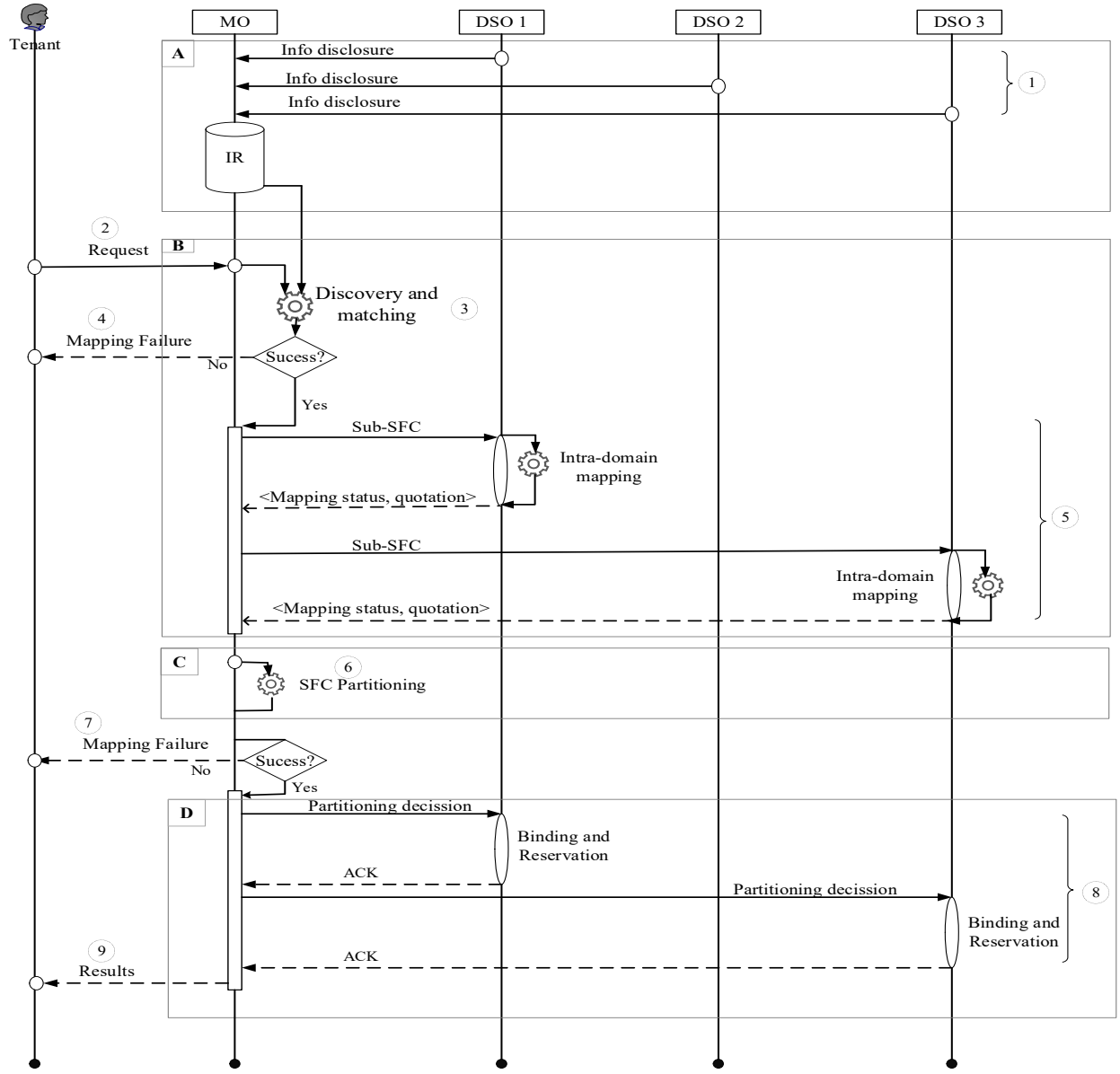



**Fig. 2** Sequence diagram of the multi-domain SFC orchestration framework

request, since in practice, such a request can only be served by a subset of all available domains. Moreover, even the potential domains may be able to serve only a portion of the request (i.e sub-SFC). Therefore, the MO exploits the resource discovery and matching algorithm (event 3) to associate to each potential InP a sub-SFC of the request that it can serve. In the event that any segment/VNF of the request has no potential candidate, then the request is rejected with a mapping failure message sent to the tenant (event 4), otherwise, the MO sends to each DSO the associated sub-SFC (event 5). Then, each DSO performs the intra-domain evaluation of the assigned sub-SFC, and sends back the intra-domain mapping results to the MO, specifying computed values of each parameter specified by the $MO$, such as: total cost, average delay, bottleneck bandwidth and guaranteed reliability within this domain. To increase competitiveness and provisioning efficiency, we propose a flexible bidding scheme, in which each DSO can return a quotation of all the possible sub-strings of the assigned sub-SFC it can provision internally. For example, instead of a given DSO returning a single aggregated quotation for a sub-SFC as $< v2, v3, v4, X >$, where X denotes the cost and other attributes, such as guaranteed reliability or delay to the different peering nodes for mapping sub-SFC with VNFs v2, v3, and v4; it instead returns the quotation for the sub-SFC components in the form of $\{< v2, X >, < v3, X >, < v4, X >$

$, < v2, v3, X >, < v3, v4, X >, < v2, v3, v4, X >\}$. This is because, even if an InP can serve a sub-SFC (or even the entire SFC), the *MO* should have the flexibility to assign any number of the feasible VNFs to be served by that InP, as long as taking such an action is more beneficial to the *MO*. Moreover, since the *DSO* will not disclose details regarding the location where the VNFs are to be embedded, the information regarding the unit cost per resource is kept private to the *DSO*, since it returns just the total quotation, without revealing the amount of resources to be incurred for mapping the request segment.

In our framework we consider the *MO* to have enough information to identify the potential candidate InPs for mapping each VNF, primarily due to location constraints. As a result, we benefit from that information to guide the different InPs about the most likely peering nodes to be used for the inter-domain connectivity (i.e.: the peering nodes connected to the candidate InPs of the preceding or subsequent VNFs). This way, the *DSO* always tries to map the assigned sub-SFC as close as possible to these peering nodes, not only to minimize the intra-domain resource consumption, but also to minimize the delay towards the exit or entry peering nodes. To understand the insight behind this guided mapping, let us consider a SFC consisting of 3 VNFs, as shown in Fig. 3. The request virtual link constraints are indicated as x/y where x and y denote the maximum delay and minimum bandwidth respectively. In the same way, the values on each substrate link indicate the delay and residual bandwidth associated with that link. As shown in the figure, in mapping solution 1, InP A maps the sub-SFC composed of VNF 1 and VNF 2 close to peering node $g$, while in mapping solution 2, these are mapped close to peering node $c$. Observe that mapping solution 1 results in a mapping failure due to the violation of the delay constraint on virtual link VNF 2 - VNF 3. As the internal topology of the domain is undisclosed, the *DSO* has no information regarding where the subsequent parts of the request will be embedded; consequently, the *DSO* may embed its allocated SFC segments in a location that is far from the peering nodes that connect to the candidate InP of the subsequent segment. In this regard, we propose that the *MO* discloses the preferred peering nodes, as it sends the sub-SFC segments to the respective candidate InPs.

In the SFC partitioning phase shown in block C, the *MO* uses the results of the candidate extraction phase to identify the final domains to host each VNF of the request (event 6). If the partitioning phase is unsuccessful, then, the *MO* sends a mapping failure message to the tenant (event 7), otherwise, the *MO*

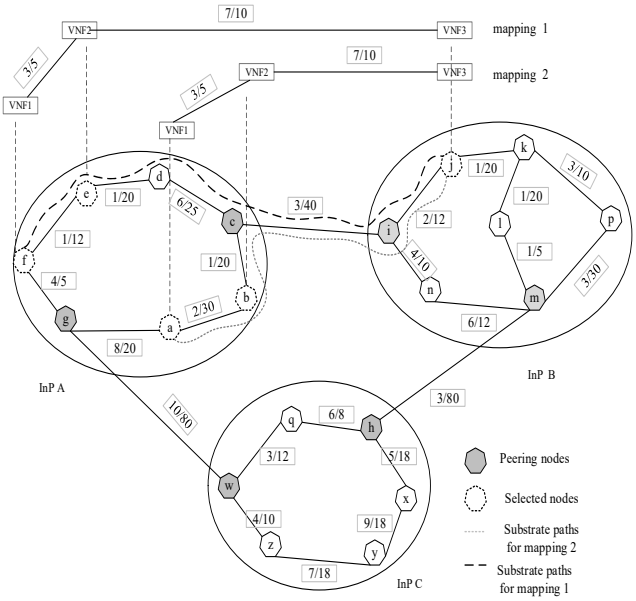



**Fig. 3** Illustration of guided mapping

instructs the selected InPs to reserve and bind the resources (event 8), including those connecting to the outgoing peering node selected by the partitioning algorithm. Observe that the partitioning algorithm jointly selects the intra-domain links to be used (by specifying the peering nodes to be used), the intra-domain nodes, and the inter-domain links. This coordinated approach guarantees a good mapping solution. Upon completing the binding step, the *MO* issues a request acceptance message to the tenant (event 9). This may be followed by the service deployment and the management of the life-cycle, including a periodic scaling of the allocated resources, depending on the real time utilization.

## 4 Service and Network modelling and Problem description

### 4.1 SFC Request

In this work, each SFC is specified as a tuple $< G_v, \tau^s, \tau^d, \tau^f >$ in which $G_v$ denotes the SFC connectivity graph modeled as a directed graph denoted by $G_v = (N_v, E_v)$. The term $N_v$ is the set of VNFs in the SFC and $E_v$ is the set of virtual links for the SFC. Each VNF $n_v \in N_v$ is characterized by: i) node resource demand denoted as $dem_c^{n_v}$ ii) function type demand denoted as $f^{n_v}$ iii) acceptable location region denoted as $r^{n_v}$. Likewise, each virtual link $e_v \in E_v$ is characterized by demanded bandwidth resource $dem_{bw}^{e_v}$. Each SFC is characterized by an end-to-end delay of $del^{\tau}$. The terms $\tau^s$, $\tau^d$ and $\tau^f$ denote the source node, destination node and lifetime of the SFC respectively.

### 4.2 Physical network

We consider a physical network comprised of M infrastructure providers. We model the physical network as a weighted undirected graph $G_s = (N_s, E_s)$ where $N_s$, $E_s$ denote the set of all physical service nodes (e.g.: virtual machines) and links respectively. For a given InP/domain m, $G_s^m = (N_s^m, E_s^m)$ is an undirected graph where $N_s^m$ and $E_s^m$ respectively denote the set of substrate nodes and intra-domain substrate links/edges that are specific to that domain. Note that $G_s^m \in G_s$, $N_s^m \in N_s$, and $L_s^m \in E_s$ . Each physical node $n_s \in N_s$ is characterized by: i) location specification $loc^{n_s}$ modeled as a point $p(x_n^s, y_n^s)$ where $x_n^s$ and $y_n^s$ are the x and y Cartesian coordinates; ii) a set of function types deployed on this node, denoted as $f^{n_s}$ iii) residual compute resources at a given time, denoted by $\omega_c^{n_s}$; iv) maximum resource capacity, denoted by $\Omega_c^{n_s}$; In addition, such a node is characterized by a unit resource price, $p^{n^s}$.

Similarly, we denote by $e_m^{kq} \in E_s^m$ as a single hop edge between physical nodes $k$ and $q$ in domain $m$. Each edge $e_m^{kq} \in E_s^m$ is characterized by: i) maximum bandwidth capacity $\Omega_{bw}^{kq}$; ii) residual bandwidth at a given time, denoted by $\omega_{bw}^{e_m^{kq}}$; iii) propagation delay $\delta(e_m^{kq})$; iv) consumed bandwidth at a given time t denoted by $Bw_t(e_m^{kq})$; and where applicable, a unit resource price $p(e_m^{kq})$.

### 4.3 Problem description

In the landscape of 5G networks and beyond, a provider can deploy network services spanning the whole infrastructure, from the mobile edge to the core network [52]. In the event that such a service cannot be served by a single InP due to reasons related to geographical location or resource constraints, the end-to-end service will be provided as a concatenation of function chains instantiated across multiple providers. As a result, deployment of such a service if not well managed, may be achieved with increased overhead information and delay, hence, severely affecting latency-sensitive services.

Within the scope of this work, the multi-domain SFC deployment problem is decomposed into three stages: Candidate evaluation stage; Splitting stage; and Binding stage. The candidate evaluation stage aims to identify all domains that can partially or wholly provision the SFC request, based on the request constraints and the intra-domain attributes. Since it may be possible that the constraints of a given VNF are satisfied by more than one InP, then, the SFC partitioning or splitting stage decides which of these feasible InPs will provision such VNFs in order to optimize a given objective, such as cost, while satisfying the end-to-end constraints imposed by the service request. Once each VNF of the request has been associated with a domain at the splitting stage, the binding stage then involves mapping the sub-SFCs inside the allocated InPs and reservation of the associated inter-domain links where necessary.

The embedding of an SFC request onto the underlying infrastructure can be defined as a mapping $M$ from the SFC graph $G_v$ to a subset of the substrate graph $G_s$, in such a way that all the constraints associated with $G_v$ are satisfied. The goal of the provisioning algorithm is not only to satisfy the SFC request constraints, but also to optimize a given objective such as mapping cost and reliability among others. In this work, the multi-domain orchestration is aimed at maximizing the long term net revenue to cost ratio received by the master orchestrator. This is achieved by minimizing the provisioning cost and the penalties associated with QoS violations due to service interruptions. The provisioning cost is the total cost of physical link and node resources used to provision a SFC request.

#### *4.3.1 Master Orchestrator objective*

By exploiting the information from the resource matching step, the master orchestrator aims to select a subset of domains and their corresponding peering nodes on which to map the request, with the goal of incurring a minimal mapping cost. Mathematically, for each admitted request, the general multi-domain orchestration problem for the $MO$ can be formulated as a cost minimization problem:

$$Minimize \quad \mathbf{C(Gv)} \tag{1}$$

where **C(Gv)** incorporates the SFC request mapping cost and the QoS violation cost. The mapping cost is directly related to the cost of node and link resources allocated to the SFC request, with the formulation as follows: Let us denote by $y_i^m \in \{0, 1\}$ a binary variable equal to one if the $MO$ assigns VNF $i$ to domain $m$, zero otherwise. We denote the corresponding cost incurred by VNF $i$ when provisioned inside domain $m$ by $c_i^m$. Whenever two adjacent VNFs $i$ and $j$ are provisioned on domains $m$ and $m'$ respectively, we denote by $P_{mm'}$ the physical path between the physical nodes on which these VNFs are provisioned, and denote by $c_{ij}^{mm'}$ the cost incurred by mapping virtual link $ij$ on such a path. Note that when $m'$ is the same as $m$, this cost corresponds to the intra-domain path cost which is part of the declared quotation of InP $m$ for mapping the Sub-SFC containing these VNFs. If $m'$ is different from $m$, the cost corresponds to the sum of intra-domain link costs towards the peering nodes of domains $m$ and

$m'$ and the inter-domain paths interconnecting these peering nodes that are traversed by virtual link $ij$. Let $y_{ij}^{mm'} \in \{0,1\}$ denote a binary variable equal to 1 if the virtual link $ij$ is provisioned on path $P_{mm'}$, zero otherwise. Consequently, the mapping cost $\boldsymbol{C_p(Gv)}$ for a given request is expressed as follows:

$$\mathbf{C_p(Gv)} = \sum_{i \in N_v} \sum_{m \in M} \boldsymbol{y_i^m c_i^m} + \sum_{P_{mm'} \in P_S} \sum_{i,j \in N_v} \boldsymbol{y_{ij}^{mm'} c_{ij}^{mm'}} \quad (2)$$

The first term in equation 2 relates to the VNF mapping costs, while the second term relates to the virtual link mapping costs. These costs are evaluated by multiplying the consumed substrate node and link resources by the corresponding unit price for the used substrate nodes and links and the duration for which these resources are used, as in equations 3 and 4:

$$\boldsymbol{c_i^m = dem_c^i \times p_c^m \times \tau^f} \quad (3)$$

where $\boldsymbol{dem_c^i}$ is the desired node resource by VNF $\boldsymbol{i}$ and $\boldsymbol{p_c^m}$ is the per unit cpu resource cost within InP $\boldsymbol{m}$ for each unit time and $\boldsymbol{\tau^f}$ denotes the lifetime of the request. Similarly, the bandwidth resource cost is calculated as:

$$\boldsymbol{c_{ij}^{mm'} = \sum_{e \in P_{mm'}} p^e dem_{bw}^{ij} \times \tau^f} \quad (4)$$

where $\boldsymbol{dem_{bw}^{ij}}$ denotes the total amount of bandwidth allocated to the virtual link $\boldsymbol{ij}$ from substrate edge $\boldsymbol{e \in P_{mm'}}$, and $\boldsymbol{p^e}$ denotes the bandwidth unit price on this edge.

On evaluating equation 1, the $\boldsymbol{MO}$ should adhere to the following constraints:

$$\boldsymbol{Bw_t^u(e) \leq \Omega_{bw}^e;\ \forall t \in T} \quad (5)$$

$$\sum_{P_{mm'} \in P_S} \sum_{e \in P_{mm'}} \boldsymbol{\delta(e) \leq del^\tau\ \forall m, m' \in M} \quad (6)$$

$$\sum_{m \in M} \boldsymbol{y_i^m = 1\ \ \forall i \in N_v} \quad (7)$$

$$\boldsymbol{y_i^m, y_{ij}^{mm'} \in \{0,1\}\ \ \forall i,j \in N_v, \forall m \in M\ \forall e \in Ps} \quad (8)$$

Constraint 5 requires the total bandwidth resources utilized from a given edge will never exceed the edge capacity. Constraint 6 requires that the total substrate delay for the entire substrate path on which the SFC is provisioned should not exceed the desired end-to-end delay of the SFC request. Constraint 7 requires that no VNF should be provisioned in more than one domain. Constraint 8 refers to the domain constraints.

On the other hand, the QoS violation cost is related to the penalties incurred by the $\boldsymbol{MO}$ due to QoS violations as a result of service interruption. In general, such a cost is evaluated as:

$$\boldsymbol{C_{QoS}(G_v) = \beta_d . f_d(Rel^r)} \quad (9)$$

where $\boldsymbol{f_d(Rel^r)}$ denotes the mapping from the end-to-end reliability of a given SFC request $\boldsymbol{Rel^r}$ to the Quality-of-Service degradation. In other-words, how a given value of service reliability translates to a given level of QoS violation. The parameter $\boldsymbol{\beta_d}$ denotes the penalty factor for each unit level of QoS violation. In this work, for simplicity, $\boldsymbol{C_{QoS}(G_v)}$ for a given request is evaluated as:

$$\boldsymbol{C_{QoS}(G_v) = \sum_{t \in T} \frac{x_k^t}{\tau^f} * rev^k} \quad (10)$$

where $\boldsymbol{x_k^t \in \{0,1\}}$ is a binary variable equal to 1 if a request $\boldsymbol{k}$ experiences a QoS violation at time t, zero otherwise, and $\boldsymbol{rev^k}$ denotes the revenue obtained from the $\boldsymbol{k^{th}}$ request throughout its lifetime and computed as shown below [53], [54]:

$$\boldsymbol{rev^k = (\sum_{i \in N_v} \gamma_c^i dem_{c^i} + \sum_{ij \in L_v} dem_{bw}^{ij} \gamma_{bw}) * \tau^f} \quad (11)$$

where $\boldsymbol{\gamma_c^i}$ is the price per unit node resource charged by the $\boldsymbol{MO}$ for serving VNF $\boldsymbol{i}$ and $\boldsymbol{\gamma_{bw}}$ is the price charged per unit bandwidth resource. $\boldsymbol{dem_{c^i}}$ and $\boldsymbol{dem_{bw}^{ij}}$ denote the demanded cpu and bandwidth resources by VNF $\boldsymbol{i}$ and virtual link $\boldsymbol{ij}$ respectively. From equation 10, the $\boldsymbol{MO}$ forfeits revenue from the tenant for all the time instants for which there is violation of QoS of the served request. In case the service experiences QoS violation throughout its life-time, the $\boldsymbol{MO}$ receives zero revenue from the tenant but pays the different InPs for the resources used to map the request, corresponding to the worst case net revenue. Therefore, in order to maximize the received net revenue/profit, the $\boldsymbol{MO}$ needs to minimize both the mapping cost and QoS violation costs.

Due to the multiple combinations of InPs to which the different VNFs of the request can be assigned, the above problem is computationally hard to solve, hence, rendering exact solutions based on solvers such as CPLEX or Gurobi unfeasible for practical delay-sensitive applications, hence, motivating the use of heuristic and meta-heuristic approaches such as that adopted in this work.

#### *4.3.2 Domain Specific Orchestrator objective*

Once the $\boldsymbol{DSO}$ has received the sub-SFC (or the entire SFC) from the $\boldsymbol{MO}$, it exploits the intra-domain

topology information at its disposal to perform the intra-domain mapping evaluation with the goal of minimizing the intra-domain mapping cost. Mathematically, this can be formulated as:

$$\boldsymbol{Minimise}\ \ \mathbf{c(gv)} \tag{12}$$

where $\boldsymbol{gv} \in \boldsymbol{G_v}$ is a sub-SFC of the request and $\mathbf{c(gv)}$ is the intra-domain cost for mapping the sub-SFC. If we denote $\boldsymbol{x^m_{i,n}} \in \mathbf{\{0,1\}}$ as a binary variable equal to 1 if VNF $\boldsymbol{i}$ is mapped on physical node $\boldsymbol{n}$ inside domain $\boldsymbol{m}$, and denote by $\boldsymbol{p_c^{m,n}}$ and $\boldsymbol{p_b^{m,e}}$ as the cost per unit of node and link resources on physical node $\boldsymbol{n}$ and physical edge $\boldsymbol{e}$ respectively, then, $\boldsymbol{c(gv)}$ in equation 12 can be expressed as:

$$\begin{aligned} \boldsymbol{c(gv)} = &\sum_{i \in N_v} \boldsymbol{dem^i_c p^{n,m}_c x^m_{i,n}} \\ &+ \sum_{ij \in E_v} \sum_{e \in E^m_s} \boldsymbol{dem^{ij}_{bw} p^{m,e}_b f^m_{ij,e}} \end{aligned} \tag{13}$$

where $\boldsymbol{f^m_{ij,e}} \in \mathbf{\{0,1\}}$ is a binary variable equal to 1 if virtual link $\boldsymbol{ij}$ is mapped onto substrate edge $\boldsymbol{e} \in \boldsymbol{E^m_s}$, where $\boldsymbol{E^m_s}$ is the set of all edges in domain $\boldsymbol{m}$. Within each domain $\boldsymbol{m} \in \boldsymbol{M}$, the objective in equation 12 is optimized under the following constraints:

Node Constraints:

$$\boldsymbol{\omega^{u,t}_c \leq \Omega^n_c}\ \ \boldsymbol{\forall t \in T, n \in N^m_s} \tag{14}$$

$$\boldsymbol{dist(n,i) \leq dev(i)}\ \boldsymbol{\forall\ i \in N_v} \tag{15}$$

$$\boldsymbol{f^i \in f^n_t}\ \boldsymbol{\forall\, i \in N_v, n \in N^m_s} \tag{16}$$

$$\sum_{n \in N^m_s} \boldsymbol{x^m_{i,n} = 1}\ \ \boldsymbol{\forall i \in N_v} \tag{17}$$

Link Constraints:

$$\boldsymbol{Bw^u_t(e^{kq}_m) \leq \Omega^e_{bw}} \tag{18}$$

$$\sum_{e \in p_{qn}} \boldsymbol{f^m_{ij,e} \delta(e) \leq \omega^{ij}_d}\ \boldsymbol{\forall i,j \in N_v, \forall m \in M} \tag{19}$$

Domain constraints:

$$\boldsymbol{f^m_{ij,e}, x^m_{i,n} \in \{0,1\}}\ \boldsymbol{\forall i,j \in N_v} \tag{20}$$

Constraint 14 is the node capacity constraint which requires that the consumed resources at any physical node should never exceed its available capacity. Constraint 15 refers to the location constraints. Constraint 16 requires that for a given VNF to be provisioned on a physical node, the resource type requirement of the VNF must be among those deployed at the physical node. Constraint 17 requires that a VNF cannot be provisioned on more than one physical node. Constraints 18, 19 and 20 denote the bandwidth resource, delay and domain constraints respectively.

## 5 Multi-domain SFC orchestration Algorithm

The key problem in multi-domain service deployment is identifying the optimal set of domains in which to orchestrate the different VNFs of a given request in order to optimise a given objective. Since each VNF may be potentially hosted by a number of domains, obtaining an optimal domain set for hosting the request is computationally intractable due to the large number of possible mapping combinations even for a small-scale network. Therefore, in this work, we propose an algorithm that is able to obtain near-optimal deployment solution in feasible time. The proposed algorithm consists of three main tasks: Candidate extraction; Request splitting; and Binding ; The candidate extraction task involves identifying the segment of the request that each InP can potentially host, including the associated cost and terms for hosting that segment. From this step, it is possible that more than one InP bids for the same request segment /VNF. Therefore, the request splitting/partitioning step is used to decide on the winner for the contested segment with the goal of optimizing the mapping objective. In this work, this step is executed by an intelligent agent implemented in form of a policy neural network. Based on the results of the splitting step, the binding task reserves both node and link resources for serving the SFC from the ingress to egress node. The execution steps of the proposed algorithm are shown in Fig. 4. On receiving a request, the ***MO*** performs resource discovery and matching to identify the possible InPs for hosting the different VNFs of this request. In-case any VNF has no potential candidate, then the request is rejected, otherwise, each possible candidate is allocated a sub-SFC of the request it can potentially host. Next, each candidate domain tries to internally bid for the assigned sub-SFC considering its internal policies and available resources and sends the results of this operation to the ***MO***. Since it is possible that multiple InPs bid for the same VNF, the ***MO*** invokes the RL-based algorithm to partition the request among the contending InPs in feasible time. A detailed description of the algorithm tasks is given below:

### 5.1 Candidate extraction

This task corresponds to block B of the sequence diagram depicted in Fig. 2, and it is aimed at extracting feasible InPs for provisioning the request. This consists of the following steps:

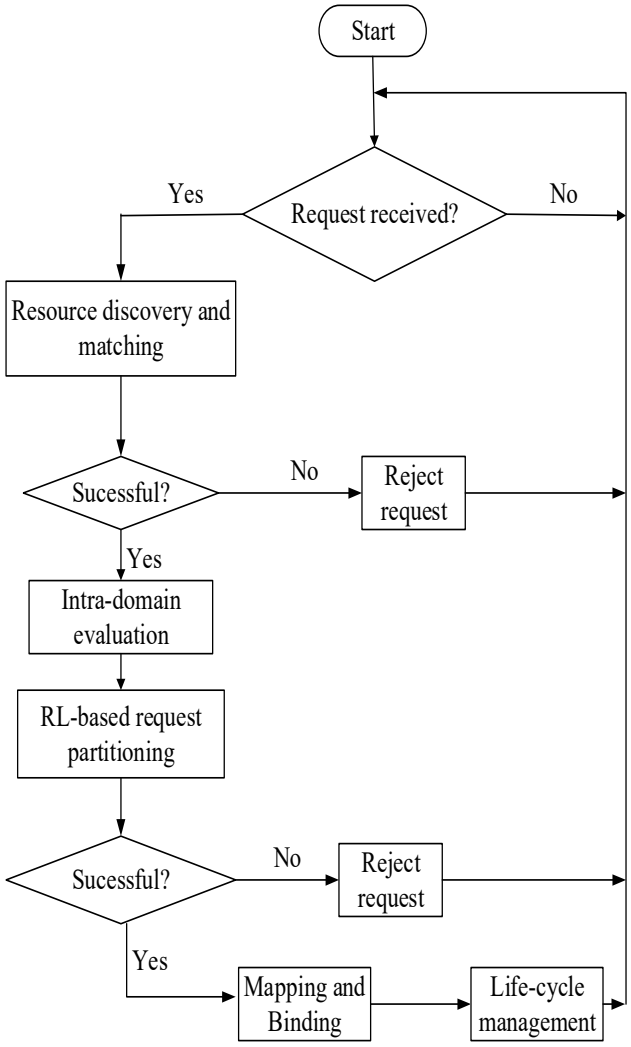



**Fig. 4** Flowchart of the multi-domain service embedding algorithm

*5.1.1 Resource discovery and matching*

The different VNFs and the interconnecting links constituting the SFC request are constrained in terms of type and amount of required resources (processing, memory, storage, bandwidth, etc.), location and delay. Consequently, each VNF and the associated virtual link(s) may be served by a subset of the available InPs. Based on the global information available at its information repository, the $\boldsymbol{MO}$ invokes the resource discovery and matching step to identify those InPs that can potentially host the request, partially or as a whole. This is done by matching the VNFs location and resource type constraints to the disclosed information of each InP, and matching the virtual links constraints to the inter-domain links' attributes. The idea behind this matching is that a VNF can only be served by an InP whose disclosed resource type set includes the required resource type of that VNF, and whose coverage satisfies the VNF location constraints. Similarly, for two successive VNFs $\boldsymbol{i}$ and $\boldsymbol{j}$ to be hosted by InP $\boldsymbol{A}$ and InP $\boldsymbol{B}$, where $\boldsymbol{A} \neg \boldsymbol{B}$, there should exist an inter-domain path between InP $\boldsymbol{A}$ and InP $\boldsymbol{B}$ that satisfies the constraints on virtual link $\boldsymbol{i-j}$. By eliminating all unfeasible InPs, the candidates set for the subsequent intra-domain enumeration is significantly reduced, resulting in a substantial decrease of the execution time of the algorithm. Specifically, the discovery and resource matching algorithm associates each VNF $\boldsymbol{j}$ of SFC $\boldsymbol{k}$ with a candidate set $\boldsymbol{cand_j^k}$ comprised of all InPs that can potentially serve this VNF. We incorporate location constraints to address cases, for example, in which the service provider may want to deploy cloud services to a set of end-users in which there could be many potential locations for a given cloud service.

*5.1.2 Intra-domain evaluation*

The resource discovery and matching step associates each InP with a sub-SFC (or SFC) that it can potentially host, based on the available global information. These sub-SFCs are then forwarded to the respective DSOs, to evaluate if and where these can be mapped, based on the intra-domain topology and policies. During this stage, each DSO will run its intra-domain mapping algorithm to evaluate the possible mapping that minimizes the mapping cost and guarantees the minimum bandwidth to the disclosed peering nodes. Once the intra-domain evaluation is completed, the DSO forwards a message to the $\boldsymbol{MO}$, indicating the portion of the assigned sub-SFC it is able to provision, the quotation (in terms of monetary cost), reliability guarantees and delay to each of the guiding peering nodes. Note that the location where the respective VNFs can be mapped, and the number of links to the peering nodes, is not disclosed. This attribute safeguards both, the topology and the pricing privacy of the InP, since the cost is returned as a single value for each sub-SFC sub-component.

Note that the intra-domain evaluation step is performed by the DSO of each InP independently and following its own policies. Therefore, any single domain SFC provisioning algorithm, such as [22],[23],[19],[24] can be used for this purpose.

## 5.2 DRL SFC request partitioning

The candidate extraction algorithm associates each VNF of the SFC with a set of InPs that can potentially host that VNF, including the possibility that a single VNF is associated with more than one InP. The SFC partitioning algorithm, therefore, is used to decide which of the

contending InPs is used to provision a given VNF in order to optimize the provisioning objective. Obtaining an optimal solution for this problem involves enumerating multiple combinations of possible mapping assignments, hence, computationally unfeasible. Moreover, the SFC provisioning objective may be jointly influenced by multiple attributes, such as cost and QoS in our case, which makes the state of the art heuristics unsuitable for this problem.

In this work, we delegate the request partitioning task to a reinforcement learning agent implemented in the form of a policy neural network. The proposed approach is able to yield a near optimal solution in a very short time. Moreover, adopting such a ML based approach enables the joint consideration of multiple attributes, since it is able to efficiently infer the influence of each attribute to the reward signal. In the sections that follow, we discuss the DRL based partitioning algorithm with special emphasis on its MDP formulation, the policy neural network architecture and its training phase.

### *5.2.1 MDP model*

The RL algorithm adopts a Markovian Decision Process (MDP) where $\boldsymbol{A}$ is the set of discrete actions, $\boldsymbol{S}$ is the set of discrete states, $\boldsymbol{P}$ is the transition probability distribution, $\boldsymbol{R}$ is the return function and $\boldsymbol{\gamma \in [0, 1]}$ is the discount factor of future rewards. In this work, we adopt a model-free method, hence, we are not interested in learning the transition probability. The return of an episode is defined for an MDP as the discounted sum of rewards received by the agent during that episode, and expressed as:

$$\boldsymbol{R = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i)} \tag{21}$$

where $\boldsymbol{r(s_i, a_i)}$ is the reward received by the agent by taking action $\boldsymbol{a_i}$ in state $\boldsymbol{s_i}$ at time instant $\boldsymbol{i}$. The goal of the RL agent, therefore, is to learn a policy $\boldsymbol{\pi : S \rightarrow A}$ which maximizes the expected return, $\mathbb{E}[\boldsymbol{R}]$ over all episodes. The state space, action space and reward of the adopted DRL-based framework is defined as below:

State space: The state captures an abstraction of the environment, and it is the basis upon which the agent decides which action to take. Therefore, taking a similar action in the same state should result in a similar reward and should transition the environment to a close next state. In our work, the state captures the features of the substrate network in relation to the requirements of the request to be partitioned. For our policy network architecture we adopt a convolutional neural network, which is well suited for image processing applications. Therefore, to conform to this requirement, we formulate the state as an image-like input using an $\boldsymbol{M \times N}$ feature matrix, where $\boldsymbol{M}$ is the number of InPs and $\boldsymbol{N}$ is the number of features extracted from each InP. The features constituting such a state matrix are discussed in section 5.2.2.

A key challenge with neural-network based architectures is that they are trained with a fixed state dimension, which makes it impractical to use such a neural-network for a different number of InPs from those used at training stage. But in fact, such a policy network should be able to work with any number of InPs. To overcome this challenge, we introduce the concept of dummy InPs with dummy feature vectors, which permits the trained policy network to be reusable even for scenarios where the number of InPs is inferior to the one used for training; therefore, avoiding the need to retrain the neural-network. In order to achieve this, we trained the policy neural-network using the maximum possible number of InPs. Then, for the testing phase, when the number of InPs is less than the one used for training, we match the input matrix by appending dummy InPs with dummy feature vectors to reach the expected state size. The dummy features are obtained by providing the worst values of each feature, in order to make such dummy InPs less likely to be selected by the policy network. Moreover, in the worst event that a dummy InP is assigned a high probability of being selected to host a given VNF, the filtering layer that we adopt at the output end of the architecture will be able to sieve out such an InP.

Action space: For each request received by the $\boldsymbol{MO}$, the number of decision steps is equal to the number of VNFs in the request. Therefore, for each VNF of the request, the policy neural-network has to decide on which InP to assign the VNF to. Therefore, the action space for each decision epoch is equal to the number of InPs in the system, including dummy InPs in the event that the number of available InPs is less than the training size. Since we have a discrete number of InPs, the action space is discrete as well, hence, well suited for the proposed RL agent.

Reward: The reward signal is aligned with the SFC deployment objective, which in this work is to maximize the long term revenue by minimizing both: the SFC mapping cost $\boldsymbol{C_P(G_v)}$; and the QoS degradation cost $\boldsymbol{C_{QoS}(G_v)}$. The SFC mapping cost captures the cost of resources used for embedding the SFC across the different InPs, as shown in equation 2. The QoS degradation cost captures the penalties resulting from the negative effects of QoS degradation due to service interruption as expressed in equation 10. In order to encourage the agent to take actions that result in low mapping cost

and high service availability, we formulate the reward signal obtained after embedding a given request as the revenue-to-cost ratio:

$$\boldsymbol{Reward} = \frac{\boldsymbol{revenue}}{\boldsymbol{Cost}} \tag{22}$$

where $\boldsymbol{Cost}$ is evaluated as the sum of the mapping cost $\boldsymbol{C_p(G_v)}$ and the QoS degradation penalty cost $\boldsymbol{C_{QoS}(G_v)}$. The revenue in this case is evaluated as the sum of the revenue from demanded cpu and bandwidth resources as below [55]:

$$\boldsymbol{revenue} = \boldsymbol{\alpha} \sum_{\boldsymbol{i \in N_v}} \boldsymbol{dem_c^i} + \boldsymbol{\beta} \sum_{\boldsymbol{ij \in E_v}} \boldsymbol{dem_{bw}^{ij}} \tag{23}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ denote the amount that the $\boldsymbol{MO}$ charges each tenant for each unit of computing and bandwidth resources respectively. In this work, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are selected as one in order to keep the reward signal between 0 and 1, so as to speed up the learning speed of the algorithm.

### *5.2.2 Feature extraction*

In order to maximize the long term reward, the policy network should select InPs that jointly minimize the request mapping cost and the failure probability of request by selecting InPs with high reliability reputation and low per unit resource value. Therefore, the Input features to the policy network are selected to reflect those attributes. The input features used for each InP, at each step, that is, for mapping the $\boldsymbol{n^{th}}$ VNF are discussed below:

• Average cost per VNF, $\boldsymbol{Cost_m^{VNF}}$: For a given InP $\boldsymbol{m}$, when mapping the $\boldsymbol{n^{th}}$ VNF, $\boldsymbol{Cost_m^{VNF}}$ is considered as the cost that InP $\boldsymbol{m}$ charges for mapping each VNF of that particular request on average, and it is computed as the cost of mapping the largest feasible sub-SFC of the present request within this InP divided by the number of VNFs within this sub-SFC. The idea behind choosing the largest Sub-SFC is to avoid greedily selecting the InP based on only the cost of mapping the current VNF. If InP $\boldsymbol{m}$ is not a candidate for the current VNF under consideration, such an InP is assigned an infinite value of $\boldsymbol{Cost_m^{VNF}}$, hence, discouraging the policy neural network from selecting that InP. The cost for the sub-SFC is part of the quotation returned to the $\boldsymbol{MO}$ after the intra-domain evaluation stage.

• Average cost of the path to the ingress and egress nodes, $\boldsymbol{Cost_m^{path}}$: For a given InP $\boldsymbol{m}$, when mapping the $\boldsymbol{n^{th}}$ VNF, this feature is computed as the cost of the inter-domain path from this InP to the ingress and egress nodes and computed as:

$$\boldsymbol{Cost_m^{path}} = \frac{\boldsymbol{Dist(m, Ing) + Dist(m, egr)}}{\boldsymbol{2}} \tag{24}$$

where $\boldsymbol{Dist(m, Ing)}$ and $\boldsymbol{Dist(m, egr)}$ denote the cost of the inter-domain path between InP $\boldsymbol{m}$ and the ingress and egress node respectively. This feature serves two purposes: First, InPs that are not reachable from the egress or ingress nodes are unfeasible for mapping a given VNF, hence, associated with infinite $\boldsymbol{Cost_m^{path}}$ value. Therefore, the probability of the policy neural network to select such InPs drastically decreases; Secondly, this feature biases the policy network towards selecting InPs with low $\boldsymbol{Cost^{path}}$ values, hence, translating in low mapping cost in terms of inter-domain paths from ingress to egress node.

• Average reliability of the InP, $\boldsymbol{Reliab_m^{VNF}}$. For a given InP $\boldsymbol{m}$, when mapping the $\boldsymbol{n^{th}}$ VNF of the request, $\boldsymbol{Reliab_m^{VNF}}$ relates to the reliability guarantee disclosed by the InP in mapping the sub-SFC containing this VNF. This value is disclosed along with the price quotation for the sub-SFC. In the event that it is not disclosed by the InP, this can be easily inferred from the previous failures and preemptions experienced from the previous mappings, to give the likelihood of the current VNF experiencing failure if hosted by InP $\boldsymbol{m}$. The goal of the policy neural network is to select InPs with high $\boldsymbol{Reliab^m}$ values for each VNF whenever possible with the target that eventually, all the VNFs of the SFC are placed on domains that can guarantee service survivability.

• Success probability $\boldsymbol{p_m^{VNF}}$. For a given InP, when mapping the $\boldsymbol{n^{th}}$ VNF of a request, $\boldsymbol{p_m^{VNF}}$ captures the fraction of the $\boldsymbol{n-1}$ previously mapped VNFs of the same request that have been mapped onto this InP. The purpose of this parameter is to encourage mapping as many VNFs as possible to the same InP as long as such an InP has low mapping cost and high reliability values. This results in a reduced number of inter-domain connections, hence the less the embedding costs, since in practice, higher costs are linked to inter-domain links. On the contrary, however, this feature may also be used to encourage the mapping of VNFs along different InPs for objectives related to load balancing or fault resilience.

### *5.2.3 Neural Network architecture*

The SFC request partitioning is performed by the MO which runs a policy neural network with the ability to learn new policies based on its environment and past decisions. The policy neural network takes as input a feature matrix which is extracted from: 1) the results of the resource discovery step (e.g.: the average cost for mapping each VNF within each InP); 2) the global information available in the information repository (e.g.: connectivity to the ingress and egress node); 3) the re-

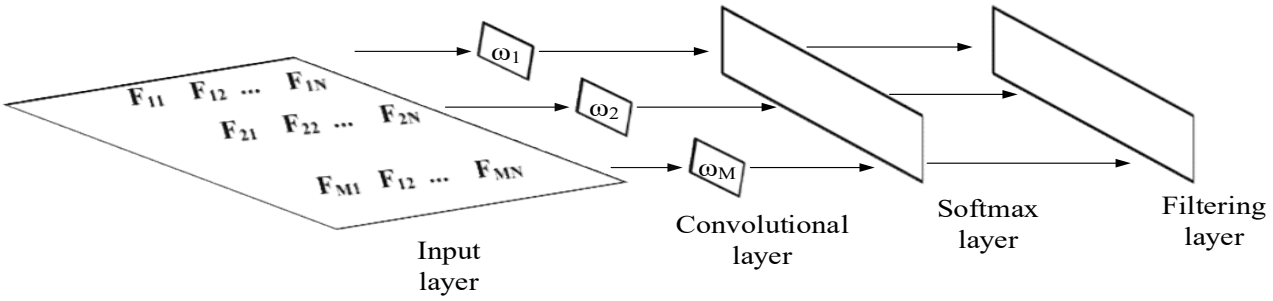



**Fig. 5** DRL Policy Neural Network Architecture

sults of the previous actions taken by this agent (e.g.: number of successful VNF mappings inside this InP). In order to make the input compatible with the convolutional network adopted in our architecture, the extracted features are converted into an image-like input, in the form of an $\boldsymbol{M} \times \boldsymbol{N}$ feature matrix, where $\boldsymbol{M}$ is the number of InPs and $\boldsymbol{N}$ is the number of features extracted from each InP, as shown in Fig. 5. The final output of the policy is a probability distribution indicating the feasibility of each InP to host the corresponding VNF, according to the desired reward function.

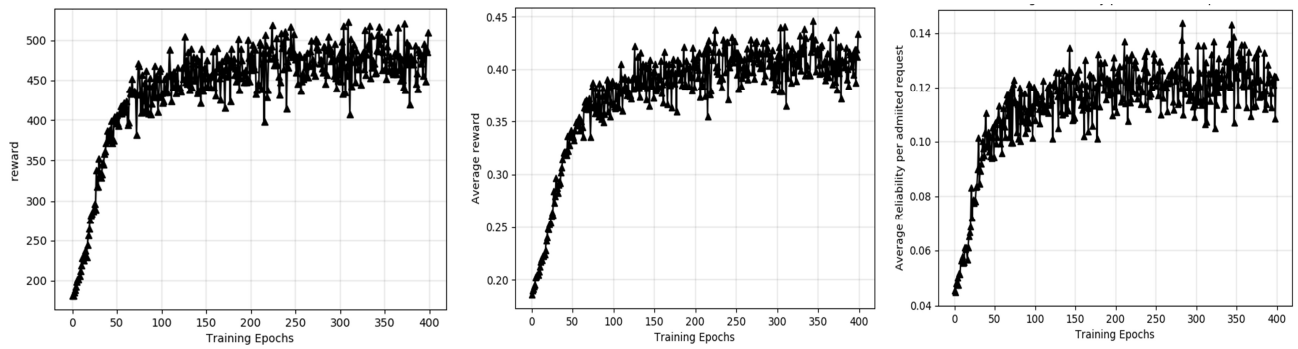
As shown in Fig. 5 our policy neural network consists of 4 layers, which are: input layer, convolutional layer, softmax layer and filtering layer. In this figure, $\boldsymbol{Fmn}$, in the input feature matrix, denotes the $\boldsymbol{n^{th}}$ feature extracted from InP $\boldsymbol{m}$. The core part of the architecture is the convolutional layer, which takes as input the feature matrix from the input layer and performs a convolution between this matrix and the learnable weight values of the filter. This operation can be seen as a dot product of each InP feature vector and the filter weights. The output of the convolutional layer is an $\boldsymbol{M-dimensional}$ vector where $\boldsymbol{M}$ is the number of considered InPs. Thus, the convolutional layer associates a single score value to each InP depending on the values of the extracted features. The score corresponding to the $\boldsymbol{m^{th}}$ InP is directly related to the suitability of that InP to host the VNF under consideration. Our motivation for using the convolutional layer is its ability to easily learn the influence of each input feature towards the desired objective. Moreover, this is achieved with minimal memory overhead compared to conventional neural network architectures.

The output from the convolutional layer is fed as input to the softmax layer, which transforms the $\boldsymbol{M-dimensional}$ vector of InP score values into an $\boldsymbol{M-dimensional}$ vector of probability distributions. The probability attached to each value relates to the probability of that InP to optimize the objective under consideration. The filtering layer is adopted at the output end of the architecture to prune those InPs that are not able to meet the request constraints. Once such InPs are filtered out, then the final candidate InP for the virtual node under consideration is chosen as the one with the highest probability value. In case of a tie between InPs for the highest probability, the candidate InP is chosen randomly from the contending InPs.

#### *5.2.4 Policy network training*

Reinforcement learning agents are able to learn optimal decisions through experience obtained by interacting with the environment, hence, able to learn even in the absence of label set. By exploiting the experience from previous actions and rewards, such an agent is able to make decisions that improve the future reward. Moreover, in the resource management domain, attributes such as traffic load characterizing the environment, are usually repetitive, with certain predictable temporal correlations. Those repetitive patterns enable the agent to learn online, as the system executes, or offline by exploiting historical information. In this work, we adopted the latter option.

In this work, the policy network was trained using a set of 1000 requests generated offline for each training episode, and the training repeated for a total of 400 episodes. The training phase considered 12 InPs as the maximum possible number of InPs in the system. For each training epoch and for each request in the demand set, the candidate evaluation step is used to generate the candidate InPs for mapping the request. Then, for each VNF of this request at a time, the corresponding feature matrix is generated which is then fed into the policy neural network. The policy network then associates a probability to each InP for embedding this VNF, inline with the mapping objective. Note, however, that since the neural network parameters are initially randomly assigned, the InP with the highest probability may not necessarily be the best InP for embedding this VNF, since the neural network accuracy is low at this stage especially for the first training episodes. This necessi-




(a) Total reward per training epoch (b) Average reward per accepted SFC request (c) Average end-to-end reliability per admitted SFC

**Fig. 6** Training Performance results

tates to perform a trade off between exploration and exploitation during the training phase as in [56].

For each selection made for a given VNF by the policy network, we use the cross-entropy as the loss function for running the back-propagation algorithm to obtain the gradients of the neural network parameters; and stack the resulting gradients. If the entire request is embedded successfully, we compute the resulting reward signal, otherwise the stacked gradients are deleted. The final gradient of the entire request is then computed as:

$$\boldsymbol{g := \alpha.r.g_s} \tag{25}$$

where $\boldsymbol{\alpha}$ is the learning rate parameter, which directly influences the learning speed and training convergence, r is the reward and $\boldsymbol{g_s}$ are the stacked gradients. The resulting gradients from the different admitted requests are stacked into a buffer. When the number of virtual requests reaches the batch size, all the gradients previously stacked are jointly applied to the model and the stack buffer is emptied. Note that whereas it is possible to perform a gradient update for each successful request individually, adopting batch processing guarantees a faster and more stable training process.
The performance of the neural network during the training phase is shown in Fig. 6 for training duration of 400 epochs.

## 6 Performance evaluation

This section describes the performance evaluation of the proposed algorithm, including the benchmark algorithms, simulation scenario , performance metrics, simulation environment, and discussion of obtained results.

### 6.1 Benchmark algorithms and simulation scenarios

The performance of the proposed multi-domain service embedding algorithm is evaluated considering both offline and online scenarios. Under the offline scenario, all demands to be served are known in advance and these are characterized by infinite life-time with respect to the simulation window. In this case, all resources allocated to a given demand cannot be released for future use by other demands. Considering this case of non-expiring demands gives a clearer insight regarding the ability of an algorithm to adapt to high loading stress as compared to the online case. In the online case, the demands continuously arrive to the system with a given distribution and with a finite life-time. Therefore, the resources assigned to a given demand are released to the system upon its expiry. For both simulation scenarios, the proposed algorithm is compared with the following algorithms:

- Distributed Network Service Embedding (DistNSE) algorithm proposed in [29]. The choice of this work for comparison is justified for a number of reasons: First, just like our work, DistNSE considers limited information disclosure and was found to be optimized in terms of mapping cost performance. Secondly, DistNSE can be easily customized to perform service deployment with the objective of enhancing service reliability. Moreover, for the fairness sake, we considered the best performance scenario of the algorithm in which all feasible paths from the ingress to the egress node are evaluated.
- Reliab-DistNSE, which uses the DistNSE algorithm above, but with the final path for embedding the service request being selected as the one which results in the highest service reliability from source to destination.
- The enhanced DistNSE ( E-DistNSE) algorithm which is the enhanced version of the DistNSE algorithm by incorporating our proposed enhancements.

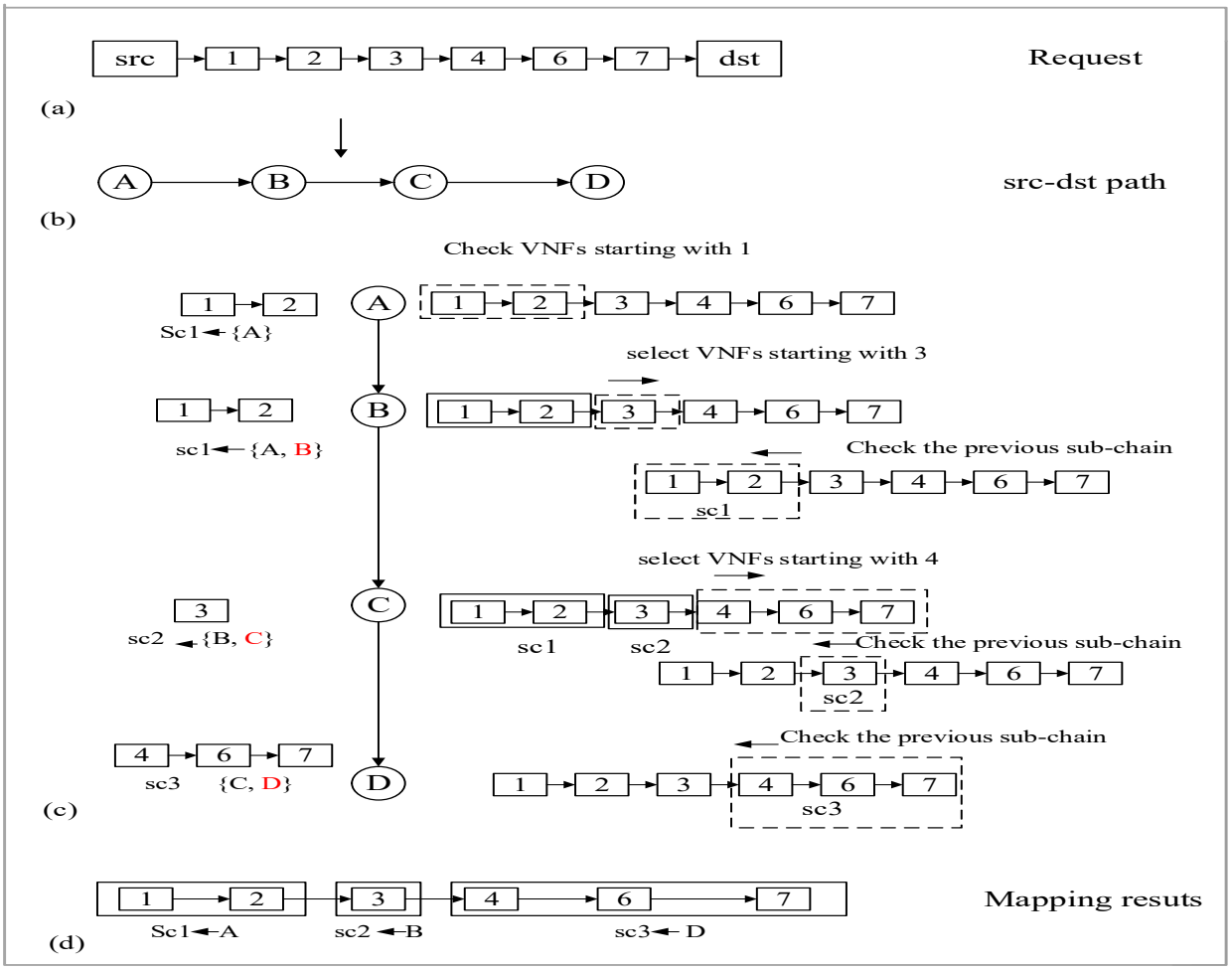



**Fig. 7** An illustration of DistNSE service provisioning

To give an insight into the enhancements made to the DistNSE algorithm, we start by giving a brief description of the DistNSE algorithm. Consider Fig. 7 in which Fig. 7(a), and 7(c) respectively show a SFC request with 7 VNFs and an illustration of the provisioning steps of the DistNSE algorithm to provision such a request. By exploiting the exposed global information such as residual bandwidth on the inter-domain links and the exposed boarder nodes of the different InPs, the algorithm starts by obtaining all feasible paths from the source node to the destination node. An example of such a path is shown in Fig. 7(b) consisting of 4 InPs . Then, for each of these paths, the algorithm execution starts by each first InP along the path (InP A in this case) receiving the SFC request to be provisioned. Then, based on its internal policy, this InP selects a sub-SFC of the request to bid for (i.e sc1 with VNFs 1,2). Next, InP A forwards the selected sub-SFC and the original SFC request to the next InP along the path (InP B), which in turn, selects a sub-SFC (sc2) from the VNFs that have not been selected. This InP also tries to compete for the sub-SFC selected by the preceding InP i.e sc1. Next, InP B forwards the selected sub-SFC and the original chain to the subsequent InP on the path where the same steps are performed. This procedure is repeated till the last InP has been reached. An example of the mapping solution from the path in Fig. 7(b) is shown in Fig. 7(d) in which the different VNFs of the request are mapped by InPs A ,B and D. Next, the mapping results for the different paths are collected, and the path with the least mapping cost is selected as the one to host the SFC request.

From the execution of this algorithm, the authors allow the InPs to compete only for the last selected sub-SFC in order to avoid violations in the service chain order. As an example, if InP C bids for sc1 and sc2 and wins sc1, whereas InP B bids for and wins sc2, the service chain order will be violated. However, this algorithm faces two main drawbacks: First, by restricting each InP to compete for only the sub-SFC selected by the preceding InP in the path, InPs at the end of the path are denied chances of mapping starting VNFs in the

request, even when these InPs can map those VNFs at a lower cost. As a result, the mapping solution of the DistNSE relies much on the position of InPs along a given path, hence, may result in a poor mapping solution. As an example, InP D can only compete for sub-SFC sc3 even when it can map the entire SFC request at a lower cost; Secondly, enumerating all possible paths from the source to destination may be time consuming for a large number of InPs. To overcome this challenge, the authors propose to set an upper limit on the maximum number of paths that can be considered. However, it is possible that paths that would result in better mapping solutions are not included in the considered paths set. To address these drawbacks, E-DistNSE incorporates two enhancements to the DistNSE algorithm as follows:

Firstly, E-DistNSE permits each InP irrespective of its position along a given path to compete for and be assigned any number of previously selected sub-SFCs as long as those sub-SFCs are consecutive and include the most recently selected sub-SFC along the path. As an example, considering Fig. 7(c), InP D can compete for sub-SFC set {sc1, sc2, sc3} or {sc2, sc3} or {sc3}. If InP D wins the first set, then the entire SFC request is mapped inside this InP. However, InP D cannot compete for {sc1, sc3} or {sc1, sc2} since the sub-SFCs in the first set are not consecutive and the second set does not include sc3, the most recent selected sub-SFC, hence, allocating such sub-SFCs to InP D would lead to violation of the order of the SFC. Note that this enhancement respects both the order of the SFC and enhances the performance of the algorithm by not restricting the number of VNFs that an InP can compete for. This makes it possible for even the last InP along the path to host the first VNFs in the request.

Secondly, as an enhancement towards the execution time of the algorithm, starting with the first path from source to destination, the minimum cost value of the cost observed so far along the path is stored. Then, for the subsequent paths, whenever the mapping cost seen so far along this path is equal to or exceeds the stored minimum cost value, the computation along this path is aborted, and the algorithm execution goes to the next path. This enhancement realizes time saving from two aspects: First, the time that would be spent on enumerating the remaining InPs along the path; Secondly, the time that would be spent on analyzing the results from all the returned paths by the central orchestrator in order to select the best path. Such an analysis would involve for-example sorting the returned paths according to cost, which tends to be time consuming as the number of returned paths grows.

### 6.2 Performance metrics

The performance of the proposed algorithms is compared with the chosen benchmark algorithms considering a number of performance metrics, including: acceptance ratio (AR), revenue to cost ratio, average cost per accepted SFC request, and the request processing time. These are commonly used metrics in the literature for assessing the performance of service embedding algorithms [1]. These parameters are explained below:

#### *6.2.1 Average acceptance ratio, AR*

The AR of a provisioning algorithm is expressed as the ratio of the embedded requests to the total requests in the system wherein, the requests in the system include both the admitted and rejected requests. Mathematically, considering the online case, the AR is evaluated as [57]:

$$\boldsymbol{AR} = \frac{\mathbf{1}}{\boldsymbol{T}} \sum_{\forall \boldsymbol{t} \in \boldsymbol{T}} \frac{\mathbf{1}}{\boldsymbol{N_D}} \times \boldsymbol{No.\ embedded\ requests} \quad (26)$$

where $\boldsymbol{T}$ denotes the total simulation window and $\boldsymbol{N_D}$ denotes the total requests in the system. In the offline case, the AR is evaluated as:

$$\boldsymbol{AR} = \frac{\boldsymbol{No.\ of\ embedded\ requests}}{\boldsymbol{N_D}} \quad (27)$$

This parameter is a direct measure of how an algorithm is able to share the underlying resources among the multiple requests in an effective manner. The embedding algorithm should guarantee a good AR performance with the constraint that there is no violation or degradation of the QoS of the admitted users.

#### *6.2.2 Average provisioning cost, APC*

Within the scope of this work, the APC relates to the total cost incurred by the $\boldsymbol{MO}$ while mapping a given SFC request. For the online case, this cost captures the sum of the mapping cost as reflected in equation 2 and the QoS violation cost as reflected in equations 10. For the offline case, since the requests are characterized by infinite lifetime with respect to the simulation window, this metric only captures the mapping cost as given in equation 2. We compute the average provisioning cost per admitted request as:

$$\boldsymbol{APC} = \frac{\mathbf{1}}{|\boldsymbol{R_A}|} \sum_{\boldsymbol{r} \in \boldsymbol{R_A}} \boldsymbol{P_c^r} \quad (28)$$

where $\boldsymbol{P_c^r}$ is the cost incurred by the $\boldsymbol{MO}$ while serving request $\boldsymbol{r} \in \boldsymbol{R_A}$, where $\boldsymbol{R_A}$ denotes the set of embedded service requests and $|\boldsymbol{R_A}|$ is the cardinality of that set.

*6.2.3 Average revenue to cost ratio, R2C*

In practice, the infrastructure provider is interested in maximizing the net revenue from mapping a given request, which can be expressed as the difference between the obtained revenue and the total cost in terms of the resources used to map this request. The commonly used performance metric for this purpose in the literature is the revenue to cost ratio. Considering the online case, the average revenue to cost ratio per admitted request is expressed as:

$$\boldsymbol{R2C} = \frac{\mathbf{1}}{|\boldsymbol{R_A}|} \sum_{\boldsymbol{r} \in \boldsymbol{R_A}} \boldsymbol{r2c^r} \tag{29}$$

where $\boldsymbol{r2c^r}$ is the revenue to cost ratio of request $\boldsymbol{r} \in \boldsymbol{R_A}$ where $\boldsymbol{R_A}$ is the set of all admitted request. The cost in this case is the provisioning cost which captures both the mapping cost and the QoS violation cost.

*6.2.4 Simulation environment and settings*

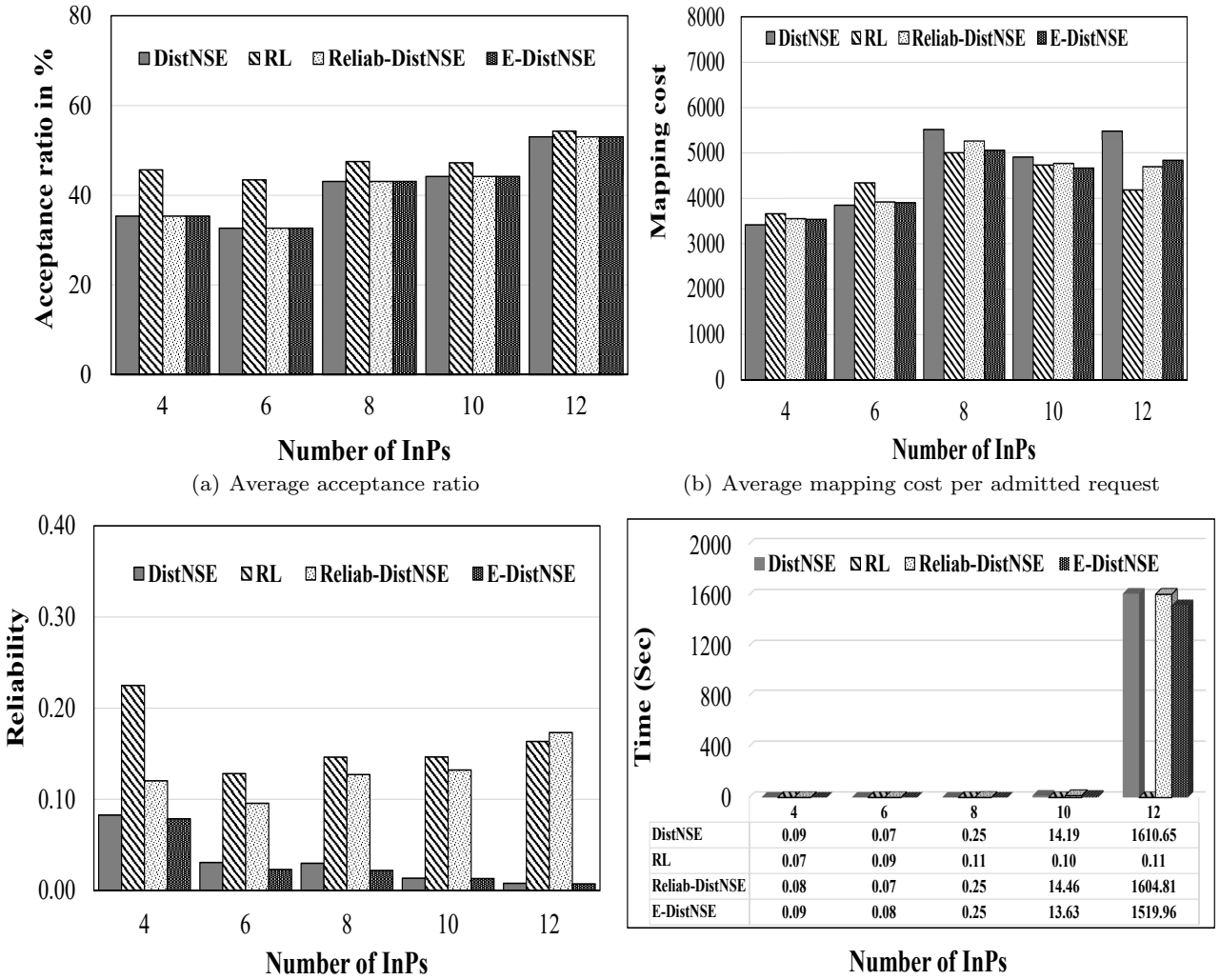
We consider a substrate network composed of InPs varied from 4 to 12, depending on the scenario, with the connection probability between InPs fixed at 0.5. The number of physical substrate nodes inside each InP is fixed at 40. The intra-domain link delay follows a uniform distribution U(1,6). The resource capacity of the intra-domain physical nodes and links follows a uniform distribution U(200,300). The above settings are similar to those adopted in [1]. The Inter-domain link resource capacity follows a uniform distribution U(400,600). The cost per unit bandwidth resource for the inter-domain links follows a uniform distribution U(10,50), while the intra-domain unit bandwidth resource cost of substrate links and unit computing resource cost of physical nodes follows a uniform distribution U(1,50). For the online case, the arrival rates follow a Poisson distribution with arrival rates varied between 2 and 14, depending on the scenario. Similarly, the life-time of such requests is exponentially distributed with a mean of 1000. The number of VNFs per SFC is varied between 3 to 15 depending on the scenario. For the online case, the service reliability of a given InP or a given inter-domain link follows a uniform distribution U(0.88, 1). This value relates to the probability that a service hosted by this InP/inter-domain link will not experience QoS violation at a given time. For the offline case, the service reliability of a given InP or a given inter-domain link follows a uniform distribution U(0.4,1). The reliability in this case relates to the probability that an offline request mapped inside this InP or inter-domain link will not experience any QoS violation throughout its life-time. Moreover, all simulations were conducted on a desktop computer running the Windows Operating System with the following features: Intel(R) Core(TM) i7-8700K CPU @ 3.70GHZ and 64GB of RAM.

6.3 Results and discussion

In this section, we analyze the performance of the proposed multi-domain service deployment algorithm considering both offline and online scenario. The proposed algorithm (RL) is compared with the benchmark algorithms discussed in section 6.1, namely, DistNSE, Reliab-DistNSE and E-DistNSE. The obtained results for the different scenarios are discussed below:

*6.3.1 Offline scenario*

In this section, we discuss the results obtained from the scenario considering offline demands. The results in Fig. 8 correspond to experiment 1 of this scenario in which the number of InPs is varied from 4 to 12. From the results in Fig. 8(a), as the number of InPs increases, the AR performance of all approaches increases. This is expected due to the increased amount of resources in the network. The proposed RL algorithm results in the highest value of AR with an average value of 47.63% while the rest of the approaches resulted in an average value of 41.66%, averaged across the different number of InPs. This is due to the fact that the RL approach jointly considers both cost minimization and reliability enhancement when mapping the request which leads to load balancing in the network, hence, avoiding link bottlenecks especially for a small number of InPs as reflected from the results. From Fig. 8(b), DistNSE results in the best performance in terms of mapping cost with low InP numbers, but this performance degrades as the number of InPs increases. This is due to the fact that DistNSE allows each InP along a given path to compete for only the sub-SFC selected by the immediate preceding InP along the path. This degrades the performance of this algorithm especially as the number of InPs along the path from source to destination increases, since the InPs at the far end of the path cannot compete for the VNFs selected by earlier InPs in the path. Overall, the RL approach results in the best performance in terms of mapping cost with an average cost of 4382.91 followed by E-DistNSE with an average cost of 4399.72 for each admitted request averaged across all substrate size. From these results, the E-DistNSE results in 5% improvement in terms of cost performance with no additional execution time compared to DistNSE whose average cost value per admitted request is 4632.69, further justifying the proposed enhancements to the DistNSE algorithm. The results in Fig. 8(c) show that the proposed RL based




(a) Average acceptance ratio

(b) Average mapping cost per admitted request

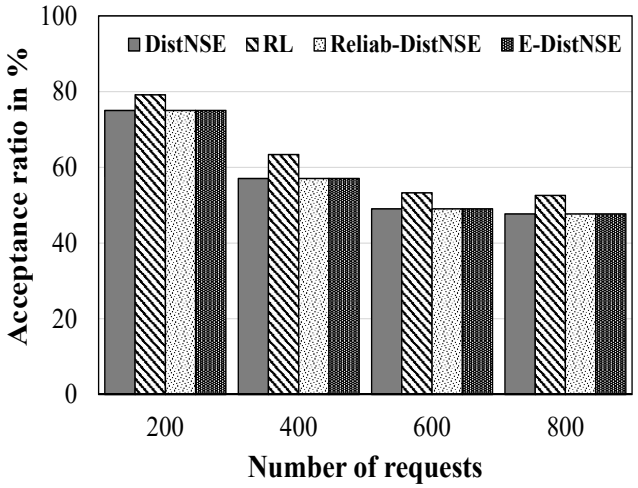
(c) Average end-to-end reliability per admitted SFC

(d) Average processing time per admitted SFC

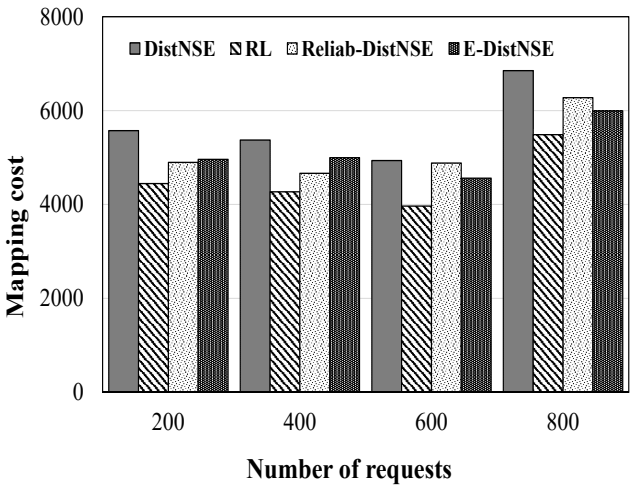
**Fig. 8** Scenario:Results of experiment 1 considering offline scenario with the number of InPs varied from 4 to 12 considering offline demand size of 3000 requests

algorithm results in the best performance in terms of service reliability with an average value of 0.163, implying that on average, each request has a 16.3% chance of not experiencing a QoS violation during its life-time. This is followed by Reliab-DistNSE, DistNSE, and E-DistNSE with average values of 0.129, 0.03 and 0.03 respectively. The good performance of the RL and Reliab-DistNSE in terms of service reliability is expected since these incorporate this parameter while mapping the request as opposed to DistNSE and E-DistNSE. The processing time of the RL algorithm is relatively constant for the different network size with an average value of 5 milliseconds for each admitted request. This is expected since the input size of the policy neutral network is constant for all the different network size. However, the time complexity of the rest of the algorithms exhibit a non-linear growth with the number of InPs. This is due to the fact that these algorithms involve computing all inter-domain paths from source to destination which is complex with a big network size.

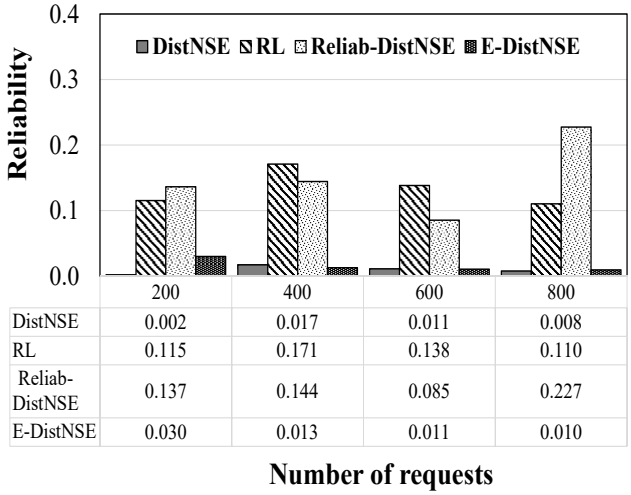
The results in Fig. 9 correspond to experiment 2 of the offline scenario in which the number of offline demands is varied from 200 to 800 demands. From the results in Fig. 9(a), the RL algorithm results in the highest performance in terms of acceptance ratio with an average value of 62.09% across the different demand size. The rest of the algorithms result in the same AR performance with an average value of 57.19%. From the results of Fig. 9(b), the RL based approach results in the lowest mapping cost with an average value of 4542.81 per admitted request across the demand different sizes. This is followed by E-DistNSE, Reliab-DistNSE and




(a) Average acceptance ratio

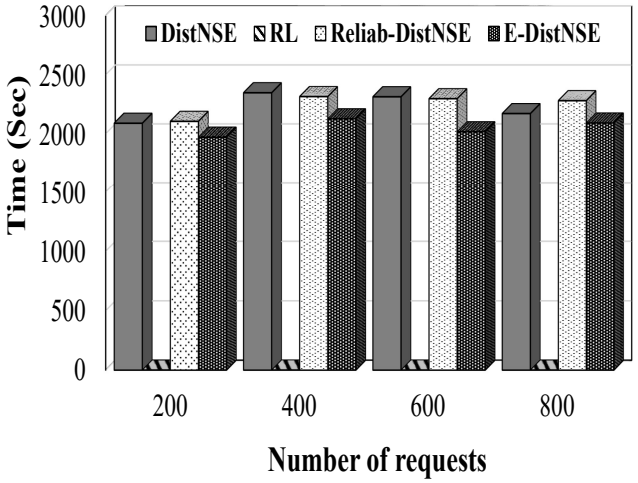



(b) Average mapping cost per admitted request




| | 200 | 400 | 600 | 800 |
|---|---|---|---|---|
| DistNSE | 0.002 | 0.017 | 0.011 | 0.008 |
| RL | 0.115 | 0.171 | 0.138 | 0.110 |
| Reliab-DistNSE | 0.137 | 0.144 | 0.085 | 0.227 |
| E-DistNSE | 0.030 | 0.013 | 0.011 | 0.010 |

(c) Average end-to-end reliability per admitted SFC



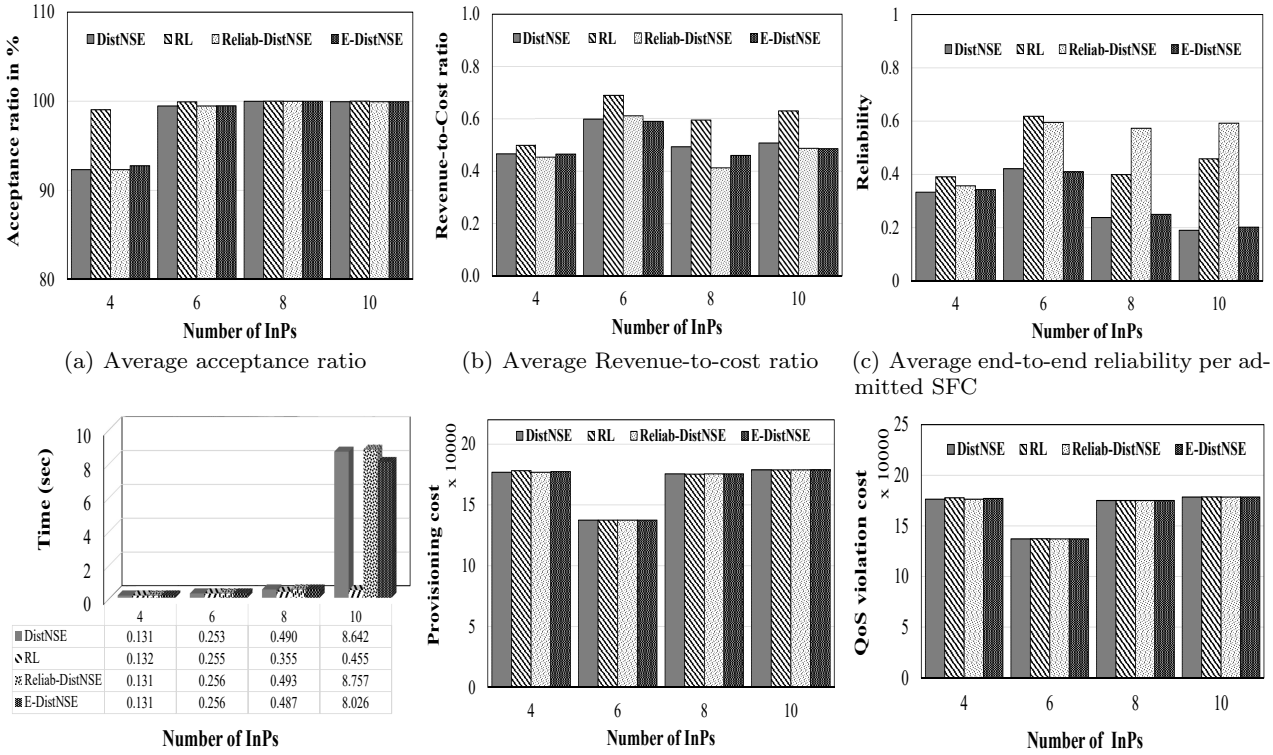

(d) Average processing time per admitted SFC

**Fig. 9** Scenario 1:Results of experiment 2 considering offline scenario with the demand size varied from 200 to 800 considering 12 InPs

DistNSE with average values of 5127.47, 5179.74 and 5683.47 respectively. These results again demonstrate that the proposed enhancements to the DistNSE algorithm results in an improvement of upto 10% in terms of mapping cost compared to the DistNSE algorithm. The results in Fig. 9(c) show that Reliab-DistNSE results in the highest service reliability with an average value of 15% followed by RL with an average value of 13% across all demand size. E-DistNSE and DistNSE results in 2% and 1% service reliability respectively.

*6.3.2 Online Scenario*

This section discusses the results obtained from the scenario considering online demands. The results of Fig. 10 correspond to experiment 1 of the online scenario in which the number of InPs is varied from 4 to 10. From Fig. 10(a), the AR performance of all algorithms improves with increase in the number of InPs as expected due to increased amount of network resources, with the RL algorithm resulting in the highest AR performance with a value of 99.74% averaged across all substrate network size. The average value of the other algorithms is 98.0%. From Fig. 10(b), the proposed RL algorithm results in the highest value of revenue-to-cost ratio with average value of 0.60 and this is followed by DistNSE, E-DistNSE and Reliab-DistNSE with average values of 0.52, 0.50 and 0.49 respectively averaged across the different InP size. The reason for the superior performance of the proposed RL algorithm in terms of revenue-to-cost ratio is due to its ability to admit requests of big size, resulting in higher revenue compared to other algorithms. This is evident from Fig. 10(f) in which the QoS violation



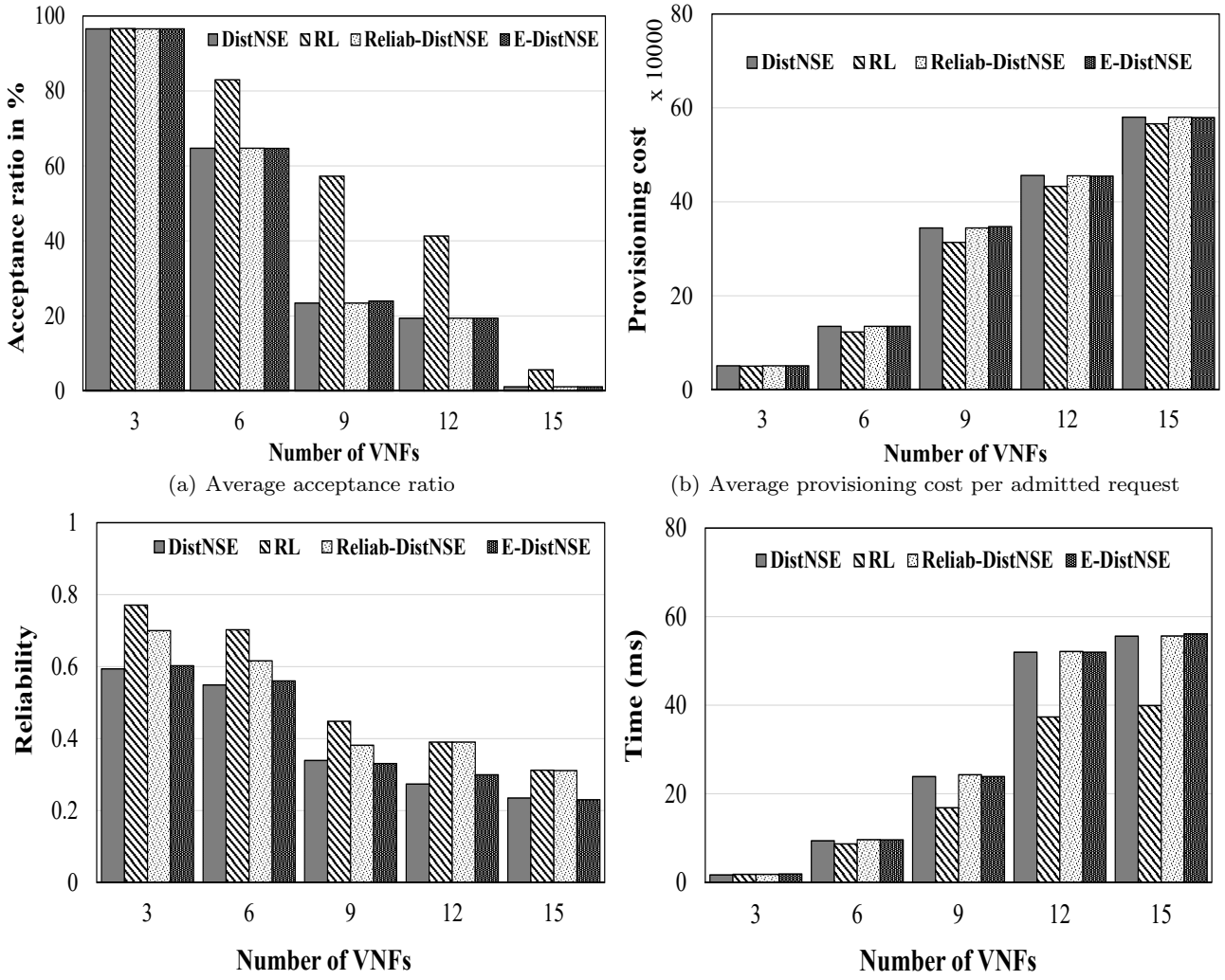

(a) Average acceptance ratio (b) Average Revenue-to-cost ratio (c) Average end-to-end reliability per admitted SFC

(d) Average processing time per admitted SFC (e) Average provisioning Cost (f) Average QoS violation cost

**Fig. 10** Experiment 1 of Online scenario in which the number of InPs is varied from 4 to 10 with the arrival rates fixed at 4 users per 100 time units for a total of 40000 time units

cost for the RL algorithm is the highest with an average value of 167147.1774 with DistNSE as an example having an average value of 166784.0736 averaged across the different InP size; yet, the average service reliability per request for the RL algorithm is 0.467 while that of DistNSE is 0.295 as shown in Fig. 10(c). This means that even with more counts of service QoS violation, the revenue forfeited by the DistNSE based algorithms is lower since the requests involved are of small size, hence, less revenue and mostly likely fewer in number (due to lower AR performance), as compared to those admitted by the RL algorithm due to its inherent load balancing attribute. From Fig. 11(c), Reliab-DistNSE and RL result in the best performance in terms of availability with average values of 0.529 and 0.467 respectively. This is expected since these incorporate service reliability attribute during the solution computation. These are followed by E-DistNSE and DistNSE with an average values of 0.30 and 0.295 respectively. As reflected in Fig. 10(d), the average execution time for all the algorithms is fraction of seconds for small numbers of InPs. However, as the number of InPs increases, all the algorithms aside from RL exhibit a drastic increase in execution time due to the paths computation strategy of these algorithms. The average execution time of the RL algorithm is 0.3 seconds followed by E-DistNSE with 2.2 seconds, and DistNSE and Reliab-DistNSE each with average value of 2.4 seconds.

The results in Fig. 11 correspond to experiment 2 of the online scenario in which the number of VNFs of each request is varied from 3 to 15. From the results of Fig. 11(a), the AR performance of all the algorithms degrades as the number of VNFs for each request increases. This is due to the fact that as the number of VNFs increases, the probability of the different VNFs finding candidate InPs to host them decreases due to the reduced amount of resources in the network, especially along the inter-domain links, since in this case, the different VNFs are more likely to be mapped across multiple InPs. The proposed RL algorithm results in the highest AR performance with an average value of 56.76% (ie 15% improvement) averaged across the different VNF numbers. The rest of the algorithms result in almost the same AR performance with an average value of 41.0%. The RL algorithm results in a superior performance, especially with high VNF numbers, due to the load balancing attribute that is inherent in this algorithm which leads to a reduction in the number of



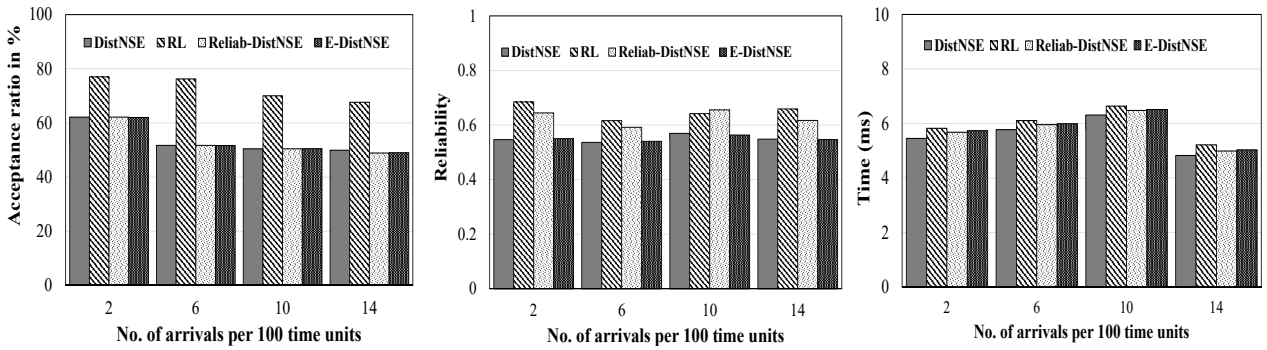

(a) Average acceptance ratio

(b) Average provisioning cost per admitted request

(c) Average end-to-end reliability per admitted SFC

(d) Average processing time per admitted SFC

**Fig. 11** Results of experiment 2 of the online scenario in which the number of VNFs per request is varied from 3 to 15 considering arrival rate of 4 users per 100 time units for a total of 40000 time units and 6 InPs




(a) Average acceptance ratio

(b) Average end-to-end reliability per admitted SFC

(c) Average processing time per admitted SFC

**Fig. 12** Results of experiment 3 of the online scenario in which the arrival rates of users is varied from 2 to 14 with the number of InPs fixed at 6

link bottlenecks in high traffic load. The results of the average provisioning cost (sum of mapping cost and QoS violation cost) are shown in Fig. 11(b), the RL algorithm results in the lowest average provisioning cost per admitted request with an average value of 297,210.28 (5.1% improvement compared to DistNSE ) followed by Reliab-DistNSE, DistNSE, and E-DistNSE with average values of 313,157.52, 313,307.63 and 313,473.73 respectively. The superior performance of RL is attributed to the fact that it considers both reliability and cost in mapping the request, hence, incurring low costs in terms of mapping and QoS violation cost. The average processing time per admitted request of all the algorithms tends to grow with the number of VNFs as reflected in Fig. 11(d). This is expected due to the intra-domain mapping overhead. However, for this case, all algorithms process each request in fractions of a second with the average processing time of the RL algorithm being 40 milliseconds while that of other algorithms is approximately 56 milliseconds. As expected, the RL and Reliab-DistNSE result in the highest service reliability with an average value of 0.52 and 0.48 respectively as reflected in Fig. 11(c). The results in Fig. 12 correspond to experiment 3 of the online scenario in which the arrival rates of the requests is varied from 2 to 14 considering 6 InPs. The results in Fig. 12(a) show that the RL algorithm results in the highest performance in terms of AR with an average value of 72.722 (26.3% improvement) with the rest of the algorithms having an average AR performance of 53.3%. The running time of all algorithms tend to increase with the number of arrivals, with all algorithms executing in a fraction of seconds for this case.

## 7 Conclusion

In this paper, we have proposed a multi-domain service deployment algorithm incorporating a reinforcement learning neural network for mapping the request. The proposed algorithm results in up-to 26% improvement in terms of acceptance ratio and up-to 10% improvement in terms of mapping and provisioning cost in some scenarios compared to the benchmark algorithms. Moreover, the proposed algorithm is found to scale with change in both network and request size. In addition, the paper has proposed an enhancement to a state of the art algorithm resulting in up-to 10% performance improvement in terms of acceptance ratio and up-to 5% improvement in terms of cost.

In this work, we have considered the requests to be characterized by fixed requirements in terms of link and node resources throughout their lifetime. However, in practice, such requirements may be characterized by temporal variations. Moreover, such variations and other network dynamics exhibit a high temporal and spatial correlation. Therefore as part of our future work, we intend to adopt a ML based approach for addressing the problem of elastic SFC deployment across multi-domain networks.

## Acknowledgment

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 777067 (NECOS project) and the national project TEC2015-71329-C2-2-R (MINECO/FEDER). This work is also supported by the " Fundamental Research Funds for the Central Universities " of China University of Petroleum (East China) under Grant 18CX02139A

## Conflict of interest

The authors declare that they have no conflict of interest.

## References

1. Sun, G., Li, Y., Liao, D., Chang, V.: Service function chain orchestration across multiple domains: A full mesh aggregation approach. IEEE Transactions on Network and Service Management **15**(3), 1175–1191 (2018). DOI 10.1109/TNSM.2018.2861717
2. Zhang, P., Yao, H., Liu, Y.: Virtual Network Embedding Based on Computing, Network, and Storage Resource Constraints. IEEE Internet of Things Journal **5**(5), 3298–3304 (2018). DOI 10.1109/JIOT.2017.2726120
3. Khebbache, S., Hadji, M., Zeghlache, D.: Virtualized network functions chaining and routing algorithms. Computer Networks **114**, 95–110 (2017). DOI 10.1016/j.comnet.2017.01.008
4. Quang, P.T.A., Bradai, A., Singh, K.D., Picard, G., Riggio, R.: Single and Multi-Domain Adaptive Allocation Algorithms for VNF Forwarding Graph Embedding. IEEE Transactions on Network and Service Management **16**(1), 98–112 (2019). DOI 10.1109/TNSM.2018.2876623
5. Sun, G., Yu, H., Anand, V., Li, L.: A cost efficient framework and algorithm for embedding dynamic virtual network requests. Future Generation Computer Systems **29**(5), 1265–1277 (2013). DOI 10.1016/j.future.2012.08.002. URL `http://dx.doi.org/10.1016/j.future.2012.08.002`
6. Kaliyammal Thiruvasagam, P., Kotagi, V.J., Murthy, C.S.R.: The More the Merrier: Enhancing Reliability of 5G Communication Services With Guaranteed Delay. IEEE Networking Letters **1**(2), 52–55 (2019). DOI 10.1109/lnet.2019.2902720
7. Zhang, P., Wang, C., Qin, Z., Cao, H.: A multidomain virtual network embedding algorithm based on multiobjective optimization for Internet of Drones architecture in Industry 4.0. Software - Practice and Experience (October 2019), 1–19 (2020). DOI 10.1002/spe.2815

8. Kibalya, G., Serrat, J., Gorricho, J.L., Yao, H., Zhang, P.: A novel dynamic programming inspired algorithm for embedding of virtual networks in future networks. Computer Networks **179**(May), 107349 (2020). DOI 10.1016/j.comnet.2020.107349. URL `https://doi.org/10.1016/j.comnet.2020.107349`
9. Kibalya, G., Serrat, J., Gorricho, J.L., Pasquini, R., Yao, H., Zhang, P.: A Reinforcement Learning Based Approach for 5G Network Slicing across Multiple Domains. In: 15th International Conference on Network and Service Management, CNSM 2019 (2019). DOI 10.23919/CNSM46954.2019.9012674
10. Leconte, M., Paschos, G.S., Mertikopoulos, P., Kozat, U.C.: A Resource Allocation Framework for Network Slicing. Proceedings - IEEE INFOCOM **2018-April**, 2177–2185 (2018). DOI 10.1109/INFOCOM.2018.8486303
11. Barakabitze, A.A., Ahmad, A., Mijumbi, R., Hines, A.: 5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges. Computer Networks **167**(November) (2020). DOI 10.1016/j.comnet.2019.106984
12. Kaur, K., Garg, S., Aujla, G.S., Kumar, N., Rodrigues, J.J., Guizani, M.: Edge Computing in the Industrial Internet of Things Environment: Software-Defined-Networks-Based Edge-Cloud Interplay. IEEE Communications Magazine **56**(2), 44–51 (2018). DOI 10.1109/MCOM.2018.1700622
13. Aujla, G.S., Kumar, N., Zomaya, A.Y., Ranjan, R.: Optimal decision making for big data processing at edge-cloud environment: An SDN perspective. IEEE Transactions on Industrial Informatics **14**(2), 778–789 (2018). DOI 10.1109/TII.2017.2738841
14. Dietrich, D., Abujoda, A., Rizk, A., Papadimitriou, P.: Multi-Provider Service Chain Embedding With Nestor. IEEE Transactions on Network and Service Management **14**(1), 91–105 (2017). DOI 10.1109/TNSM.2017.2654681
15. Dietrich, D., Rizk, A., Papadimitriou, P.: Multi-Provider Virtual Network Embedding With Limited Information Disclosure. IEEE Transactions on Network and Service Management **12**(2), 188–201 (2015). DOI 10.1109/TNSM.2015.2417652
16. Samuel, F., Chowdhury, M., Boutaba, R.: PolyViNE: Policy-based virtual network embedding across multiple domains. Journal of Internet Services and Applications **4**(1), 1–23 (2013). DOI 10.1186/1869-0238-4-6
17. Afolabi, I., Bagaa, M., Taleb, T., Flinck, H.: End-To-end network slicing enabled through network function virtualization. 2017 IEEE Conference on Standards for Communications and Networking, CSCN 2017 pp. 30–35 (2017). DOI 10.1109/CSCN.2017.8088594
18. Shahriar, N., Ahmed, R., Chowdhury, S.R., Khan, A., Boutaba, R., Mitra, J.: Generalized recovery from node failure in virtual network embedding. IEEE Transactions on Network and Service Management **14**(2), 261–274 (2017). DOI 10.1109/TNSM.2017.2693404
19. Ding, W., Yu, H., Luo, S.: Enhancing the reliability of services in NFV with the cost-efficient redundancy scheme. IEEE International Conference on Communications **1**, 1–6 (2017). DOI 10.1109/ICC.2017.7996840
20. Beck, M.T., Botero, J.F., Samelin, K.: Resilient allocation of service Function chains. 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks, NFV-SDN 2016 pp. 128–133 (2017). DOI 10.1109/NFV-SDN.2016.7919487
21. Cotroneo, D., De Simone, L., Iannillo, A.K., Lanzaro, A., Natella, R., Fan, J., Ping, W.: Network function virtualization: Challenges and directions for reliability assurance. Proceedings - IEEE 25th International Symposium on Software Reliability Engineering Workshops, ISSREW 2014 pp. 37–42 (2014). DOI 10.1109/ISSREW.2014.48
22. Sun, G., Xu, Z., Yu, H., Chen, X., Chang, V., Vasilakos, A.V.: Low-latency and Resource-efficient Service Function Chaining Orchestration in Network Function Virtualization. IEEE Internet of Things Journal **PP**(c), 1–1 (2019). DOI 10.1109/jiot.2019.2937110
23. Sun, G., Zhu, G., Liao, D., Yu, H., Du, X., Guizani, M.: Cost-Efficient Service Function Chain Orchestration for Low-Latency Applications in NFV Networks. IEEE Systems Journal **13**(4), 3877–3888 (2019). DOI 10.1109/JSYST.2018.2879883
24. Zhang, P., Yao, H., Qiu, C., Liu, Y.: Virtual network embedding using node multiple metrics based on simplified electre method. IEEE Access **6**, 37314–37327 (2018). DOI 10.1109/ACCESS.2018.2847910
25. Di Mauro, M., Longo, M., Postiglione, F.: Availability Evaluation of Multi-tenant Service Function Chaining Infrastructures by Multidimensional Universal Generating Function. IEEE Transactions on Services Computing pp. 1–13 (2018). DOI 10.1109/TSC.2018.2885748
26. Sun, G., Liao, D., Zhao, D., Sun, Z., Chang, V.: Towards provisioning hybrid virtual networks in federated cloud data centers. Future Generation Computer Systems **87**, 457–469 (2018). DOI 10.1016/j.future.2017.09.065. URL `https://doi.org/10.1016/j.future.2017.09.065`
27. Houidi, I., Louati, W., Ben Ameur, W., Zeghlache, D.: Virtual network provisioning across multiple substrate networks. Computer Networks **55**(4), 1011–1023 (2011). DOI 10.1016/j.comnet.2010.12.011
28. Zhang, Q., Wang, X., Kim, I., Palacharla, P., Ikeuchi, T.: Vertex-centric computation of service function chains in multi-domain networks. IEEE NETSOFT 2016 - 2016 IEEE NetSoft Conference and Workshops: Software-Defined Infrastructure for Networks, Clouds, IoT and Services pp. 211–218 (2016). DOI 10.1109/NETSOFT.2016.7502415
29. Abujoda, A., Papadimitriou, P.: DistNSE: Distributed network service embedding across multiple providers. 2016 8th International Conference on Communication Systems and Networks, COMSNETS 2016 (i), 1–8 (2016). DOI 10.1109/COMSNETS.2016.7439948
30. Yao, H., Chen, X., Li, M., Zhang, P., Wang, L.: A novel reinforcement learning algorithm for virtual network embedding. Neurocomputing **284**, 1–9 (2018). DOI 10.1016/j.neucom.2018.01.025. URL `https://doi.org/10.1016/j.neucom.2018.01.025`
31. Mwanje, S.S., Schmelz, L.C., Mitschele-Thiel, A.: Cognitive Cellular Networks: A Q-Learning Framework for Self-Organizing Networks. IEEE Transactions on Network and Service Management **13**(1), 85–98 (2016). DOI 10.1109/TNSM.2016.2522080
32. Moysen, J., Giupponi, L.: From 4G to 5G: Self-organized network management meets machine learning. Computer Communications **129**(January), 248–268 (2018). DOI 10.1016/j.comcom.2018.07.015. URL `https://doi.org/10.1016/j.comcom.2018.07.015`
33. Zhang, P., Wang, C., Jiang, C., Benslimane, A.: Security-Aware Virtual Network Embedding Algorithm based on Reinforcement Learning. IEEE Transactions on Network Science and Engineering **4697**(c), 1–11 (2020). DOI 10.1109/TNSE.2020.2995863
34. Singh, M., Aujla, G.S.S., Singh, A., Kumar, N., Garg, S.: Deep Learning based Blockchain Framework for Secure Software Defined Industrial Networks. IEEE Transactions

on Industrial Informatics **3203**(c), 1–1 (2020). DOI 10.1109/tii.2020.2968946
35. Houidi, I., Louati, W., Zeghlache, D.: A distributed virtual network mapping algorithm. IEEE International Conference on Communications pp. 5634–5640 (2008). DOI 10.1109/ICC.2008.1056
36. Mahmoodi, T., Van Helvoort, H., Mansfield, S.: Management and orchestration. IEEE Communications Standards Magazine **1**(4), 60 (2017). DOI 10.1109/MCOMSTD.2017.8258603
37. Shen, M., Xu, K., Yang, K., Chen, H.H.: Towards efficient virtual network embedding across multiple network domains. IEEE International Workshop on Quality of Service, IWQoS pp. 61–70 (2014). DOI 10.1109/IWQoS.2014.6914301
38. Dietrich, D., Rizk, A., Papadimitriou, P.: Multi-domain virtual network embedding with limited information disclosure. 2013 IFIP Networking Conference, IFIP Networking 2013 **12**(2), 188–201 (2013)
39. Cao, H., Zhu, Y., Yang, L., Zheng, G.: A Efficient Mapping Algorithm with Novel Node-Ranking Approach for Embedding Virtual Networks. IEEE Access **5**, 22054–22066 (2017). DOI 10.1109/ACCESS.2017.2761840
40. Li, S., Saidi, M.Y., Chen, K.: Multi-domainvirtualnetwork embedding with coordinated link mapping. Advances in Science, Technology and Engineering Systems **2**(3), 545–552 (2017). DOI 10.25046/aj020370
41. Martin-Perez, J., Bernardos, C.J.: Multi-Domain VNF Mapping Algorithms. IEEE International Symposium on Broadband Multimedia Systems and Broadcasting, BMSB **2018-June** (2018). DOI 10.1109/BMSB.2018.8436765
42. Xu, Q., Gao, D., Zhou, H., Quan, W., Shi, W.: An energy-aware method for multi-domain service function chaining. Journal of Internet Technology **19**, 1727–1739 (2018). DOI 10.3966/160792642018111906010
43. Boutigny, F., Betgé-Brezetz, S., Debar, H., Blanc, G., Lavignotte, A., Popescu, I.: Multi-provider secure virtual network embedding. 2018 9th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2018 - Proceedings **2018-Janua**, 1–5 (2018). DOI 10.1109/NTMS.2018.8328706
44. Sun, G., Li, Y., Zhu, G., Liao, D., Chang, V.: Energy-efficient service function chain provisioning in multi-domain networks. IoTBDS 2018 - Proceedings of the 3rd International Conference on Internet of Things, Big Data and Security **2018-March**(January), 144–163 (2018). DOI 10.5220/0006770301440152
45. Sun, G., Li, Y., Yu, H., Vasilakos, A.V., Du, X., Guizani, M.: Energy-efficient and traffic-aware service function chaining orchestration in multi-domain networks. Future Generation Computer Systems **91**, 347–360 (2019). DOI 10.1016/j.future.2018.09.037. URL `https://doi.org/10.1016/j.future.2018.09.037`
46. Carlo, M., Search, T., Haeri, S., Member, S., Trajkovi, L.: Virtual Network Embedding via **48**(2), 1–12 (2016)
47. Dolati, M., Hassanpour, S.B., Ghaderi, M., Khonsari, A.: DeepViNE: Virtual Network Embedding with Deep Reinforcement Learning. INFOCOM 2019 - IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2019 pp. 879–885 (2019). DOI 10.1109/INFCOMW.2019.8845171
48. Zhang, Z., Su, S., Lin, Y., Cheng, X., Shuang, K., Xu, P.: Adaptive multi-objective artificial immune system based virtual network embedding. Journal of Network and Computer Applications **53**, 140–155 (2015). DOI 10.1016/j.jnca.2015.03.007. URL `http://dx.doi.org/10.1016/j.jnca.2015.03.007`
49. Pham, T.A.Q., Hadjadj-aoul, Y., Outtagarts, A.: VNF-FG embedding: A deep reinforcement learning approach. IEEE Transactions on Network and Service Management pp. 1–10 (2019)
50. Baggio, G., Francescon, A., Fedrizzi, R.: Multi-domain service orchestration with X-MANO. 2017 IEEE Conference on Network Softwarization: Softwarization Sustaining a Hyper-Connected World: en Route to 5G, NetSoft 2017 pp. 5–6 (2017). DOI 10.1109/NETSOFT.2017.8004259
51. Francescon, A., Baggio, G., Fedrizzi, R., Grida, I., Yahia, B., Riggio, R.: X – MANO : Cross – domain Management and Orchestration of Network Services (July) (2017)
52. Tusa, F., Clayman, S., Valocchi, D., Galis, A.: Multi-Domain Orchestration for the Deployment and Management of Services on a Slice Enabled NFVI. 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks, NFV-SDN 2018 pp. 1–5 (2018). DOI 10.1109/NFV-SDN.2018.8725769
53. Cao, H., Chen, J., Guo, Y., Zhu, H., Yang, L.: A novel and one-stage embedding algorithm for mapping virtual networks. 2018 24th Asia-Pacific Conference on Communications, APCC 2018 pp. 156–161 (2019). DOI 10.1109/APCC.2018.8633537
54. Bianchi, F., Presti, F.L.: A markov reward based resource-latency aware heuristic for the virtual network embedding problem. Performance Evaluation Review **44**(4), 57–68 (2017). DOI 10.1145/3092819.3092827
55. Gong, L., Wen, Y., Zhu, Z., Lee, T.: Toward profit-seeking virtual network embedding algorithm via global resource capacity. Proceedings - IEEE INFOCOM pp. 1–9 (2014). DOI 10.1109/INFOCOM.2014.6847918
56. Amiri, R., Mehrpouyan, H., Fridman, L., Mallik, R.K., Nallanathan, A., Matolak, D.: A Machine Learning Approach for Power Allocation in HetNets Considering QoS. IEEE International Conference on Communications **2018-May** (2018). DOI 10.1109/ICC.2018.8422864
57. Hejja, K., Hesselbach, X.: Online power aware coordinated virtual network embedding with 5G delay constraint. Journal of Network and Computer Applications **124**(February), 121–136 (2018). DOI 10.1016/j.jnca.2018.10.005. URL `https://doi.org/10.1016/j.jnca.2018.10.005`